UCRL-MA-118543 Pt 3

The Basis System, Part III

EZN User Manual

Written by

Fred N. Fritsch
Yu-Hsing Chiu
Paul F. Dubois

December 22, 2000

COPYRIGHT NOTICE

All files in the Basis system are Copyright 1994-2000, by the Regents of the University of California. All rights reserved. This work was produced at the University of California, Lawrence Livermore National Laboratory (UC LLNL) under contract no. W-7405-ENG-48 (Contract 48) between the U.S. Department of Energy (DOE) and The Regents of the University of California (University) for the operation of UC LLNL. Copyright is reserved to the University for purposes of controlled dissemination, commercialization through formal licensing, or other disposition under terms of Contract 48; DOE policies, regulations and orders; and U.S. statutes. The rights of the Federal Government are reserved under Contract 48 subject to the restrictions agreed upon by the DOE and University as allowed under DOE Acquisition Letter 88-1.

DISCLAIMER

This software was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

DOE Order 1360.4A Notice

This computer software has been developed under the sponsorship of the Department of Energy. Any further distribution by any holder of this software package or other data therein outside of DOE offices or other DOE contractors, unless otherwise specifically provided for, is prohibited without the approval of the Energy, Science and Technology Software Center. Requests from outside the Department for DOE-developed computer software shall be directed to the Director, ESTSC, P.O. Box 1020, Oak Ridge, TN, 37831-1020.

Contents

CHAPTER 1: The Basis System 1

1.1 Environment Variables 1
1.2 Basis Is Both a Program and a Development System 1
1.3 About This Manual 2

CHAPTER 2: Introduction to EZN 3

2.1 Essential Setups and Simple Experiments 3
2.2 Incorporating EZN in your program 13

CHAPTER 3: Devices 15

3.1 Device Commands 15
3.2 CGM File Output 17
3.3 Working with Windows 18
3.4 Setting the Background Color 20
3.5 Setting the Colormap 20

CHAPTER 4: The EZN Graphics Model 23

4.1 The Additive Model 23
4.2 Controlling Layout 23
4.3 Plot Command Summary 25

CHAPTER 5: Attributes 27

5.1 Attribute Types 27
5.2 attr: Setting Attributes 29
5.3 Attribute Table 31

CHAPTER 6: General Plot Commands 35

6.1 plot: Plotting Curves and Markers 35
6.2 plotz: Plotting Contours 39
6.2.1 Contour Levels 40
6.2.2 Contour Control Parameters 40
6.2.3 Contour Color Fill 41

6.2.4 Contour Level Annotations 41
6.3 ploti: Cell Array Plots 44
6.3.1 Color-Mapping Functions 44

CHAPTER 7: Mesh-Oriented Commands. 47

7.1 plotm: Plotting Meshes, Boundaries, and Regions 48
7.2 plotc: Plotting Contours 54
7.3 plotf: Fillmesh Plot 57
7.3.1 Fillmesh Level Annotation. 58
7.3.2 Color-Mapping Functions 58
7.4 plotv: Plotting Vectors 62
7.5 plotr: Lasnex Rayplots 66

CHAPTER 8: Polygonal-Mesh Commands 69

8.1 plotp: Plotting Polygonal Meshes 69
8.2 plotpf: Polygonal Fillmesh Plot 74

CHAPTER 9: Surface Plot Commands 75

9.1 srfplot: 3-D Surface Plot 75
9.2 isoplot: 3-D Isosurface Plot 77

CHAPTER 10: Frame Control 79

10.1 frame: Set Frame Limits 79
10.2 nf: New Frame 83
10.3 sf: Show Frame 84
10.4 undo: Undo a Plot Command. 84

CHAPTER 11: Axes, Titles and Text 85

11.1 Changing Autograph Parameters 85
11.2 titles: Put Titles on a Plot 86
11.3 text: Put Text in the Interior of a Plot 87
11.4 ftext: Put Text Anywhere in a Frame 89
11.5 Text Quality and Optional Fonts 90

CHAPTER 12: Stream Output to Graphics 91

CHAPTER 13: Quadrant Mode 93

CHAPTER 14: Interactive Graphics Tools 97

14.1 General Graphics Applications 97
14.1.1 Zoom 97
14.1.2 Unzoom 97
14.2 Lasnex-Specific Applications 98
14.2.1 Marking Points 98
14.2.2 Marking Mesh Lines 98
14.2.3 Marking Mesh Segments 98
14.2.4 Marking Regions 99
14.2.5 Marking Zones 99

CHAPTER 15: Control Variables and Defaults 101

15.1 EZN Control Variables 101
15.1.1 Ezcurve Variables 101
15.1.2 Device Control Variables 103
15.1.3 Ezcurve Default Variables 104
15.2 Parameter Access Routines 105
15.2.1 Query EZN Parameters 105
15.2.2 Set EZN Parameters 105
15.2.3 Query and Set NCAR Parameters 106

CHAPTER 1: The Basis System

1.1 Environment Variables

Before using Basis, you should set some environment variables as follows.

- `BASIS_ROOT` should contain the name of the root of your Basis installation, `/usr/local/basis` for example.
- `MANPATH` should contain a component `$BASIS_ROOT/man`.
- Your path should contain a component `$BASIS_ROOT/bin`.
- `DISPLAY` should contain the name of your X-Windows display.
- `NCARG_ROOT` should contain the name of the root directory of your NCAR 3.2 or later distribution, if you have it.
- `gksdir` should contain the library directory for ATC-GKS, if you have it.

Check with your System Manager for the exact specifications on your local systems.

1.2 Basis Is Both a Program and a Development System

Basis is a system for developing interactive computer programs in Fortran, with some support for C and C++ as well. Using Basis you can create a program that has a sophisticated programming language as its user interface so that the user can set, calculate with, and plot, all the major variables in the program. The program author writes only the scientific part of the program; Basis supplies an environment in which to exercise that scientific programming which includes an interactive language, an interpreter, graphics, terminal logs, error recovery, macros, saving and retrieving variables, formatted I/O, and on-line documentation.

`basis` is the name of the program which results from loading the Basis System with no attached physics. It is a useful program for interactive calculations and graphics. Authors create other programs by specifying one or more packages of variables and modules to be loaded. A package is specified using a Fortran source and a variable description file in which the user specifies the common blocks to be used in the Fortran source and the functions or subroutines that are to be callable from the interactive language parser.

Basis programs are *steerable applications*, that is, applications whose behavior can be greatly modified by their users. Basis also contains optional facilities to help authors do their jobs more easily. A library of Basis packages is available that can be added to a program in a few seconds. The progammable nature of the application simplifies testing and debugging.

The Basis Language includes variable and function declarations, graphics, several looping and conditional control structures, array syntax, operators for multiplication, dot product, transpose, array or character concatenation, and a stream I/O facility. Data types include real, double, integer, complex, logical, character, chameleon, and structure. There are more than 100 built-in functions, including all the Fortran intrinsics.

Basis' interaction with compiled routines is particularly powerful. When calling a compiled routine from the interactive language, Basis verifies the number of arguments and coerces the types of the actual arguments to match those expected by the function. A compiled function can also call a user-defined function passing arguments through common.

1.3 About This Manual

This manual is part of a series of manuals documenting the Basis System. They are:

- I. Running A Basis Program: A Tutorial For Beginners
- II. Basis Language Reference Manual
- III. EZN User Manual
- IV. EZD Graphics Device Library
- V. Writing Basis Programs: A Manual For Program Authors
- VI. Basis Library Manual

The first three parts form a basic document set for a user of programs written with Basis. The remainder form a document set for an author of such programs.

Basis is available on Cray machines under the UNICOS operating system and on SUN, HP, SGI, DEC, and IBM workstations. A Linux port has been tested on at least the RedHat and Debian distributions, Intel Pentium hardware. The Portland Group Fortran <www.pgroup.com> compiler is required.

A great many people have helped create Basis and its documentation. The principal author is Paul Dubois. Major contributors are Peter Willmann, Janet Takemoto, Susan Taylor, Bruce Langdon, Jim Crotinger, Sharon Wilson, Lee Busby, Yu-Hsing Chiu, Zane Motteler, Bert Still, Robyn Allsman, Barbara Dubois, David Sinck, Kelly Barrett, and Fred Fritsch. The authors of this manual stand as representative of their efforts and those of a much larger number of minor contributors.

Send any comments about these documents to "support@icf.llnl.gov" on the Internet or to "support" on Lasnet.

CHAPTER 2: Introduction to EZN

2.1 Essential Setups and Simple Experiments

EZN is a Basis package which supplies a user interface to the National Center of Atmospheric Research (NCAR) Graphics library (see http://ngwww.ucar.edu). The EZN package is a standard part of the programs `Lasnex` and `Sod`. The EZN package has an additive model, that is, each plot command you issue adds something to the picture until you issue a `nf` ("new frame") command. The package contains curve, marker, contour, and text commands with facilities for titles, frame control, and viewport control. It also contains some commands which use Lasnex-specific data structures, such as mesh and mesh-based contour commands.

EZN will work with either the Graphics Kernel System (GKS) which comes with NCAR (which makes Computer Graphic Metafile (CGM) files in the NCAR format and supports XWindows for version NCAR3.2 or later distribution) or with Advanced Technology Center (ATC)-GKS, which supports a variety of devices. The current version of the package (in Basis 11.8) requires NCAR4.0.1 or later if NCAR-GKS is used.

Before using a program containing EZN, make sure the Basis environment variables are set correctly. Refer to Section 1.1 "Environment Variables" in Chapter 1 for the list of environment variables needed.

The most effective way to learn EZN is by doing some simple experiments. If you are executing one of the programs containing the EZN package (Basis, Sod, etc.), you can enter the following examples at your terminal to see how EZN works. Any text following a "#" is a comment. We set the variable `ezcshow` to `false` so that the pictures will not appear until the `nf` command is given. If you want to see the picture before it is completed, you can issue a `sf` or "show frame" command at any time.

```
ezcshow=false    #Don't show pictures until "nf" command given.
win on           # Open an Xwindow on your workstation.
cgm on           # Open a CGM file to record the pictures.
```

The following statements set up values for variables used in later commands.

```
# Calculate some data to be used in the examples:
real x(20)=iota(20)
real y1=x**2, y2=x**2.1, y3=x**2.2
```

```
# Example 1
titles "Example 1","One Curve","Y1"
plot y1 x
sf      # Show the frame.
```

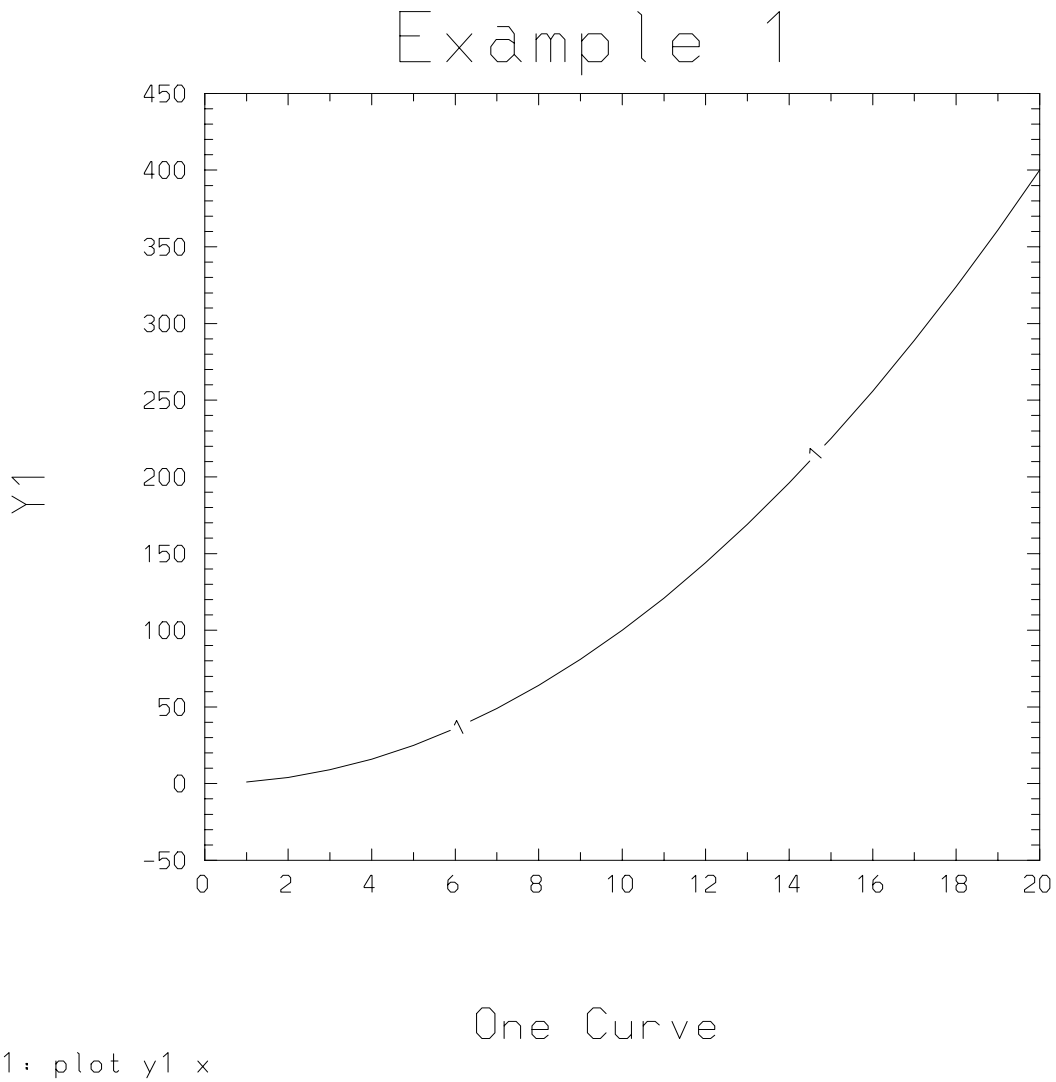


Figure 2.1: Example 1

```
# Example 2
# Change the thickness, color of the plots.
nf      # Clear display list for next example.
titles "Example 2","One Curve, Thick and in Red","Y1"
plot y1 x color=red thick=2.
sf
```

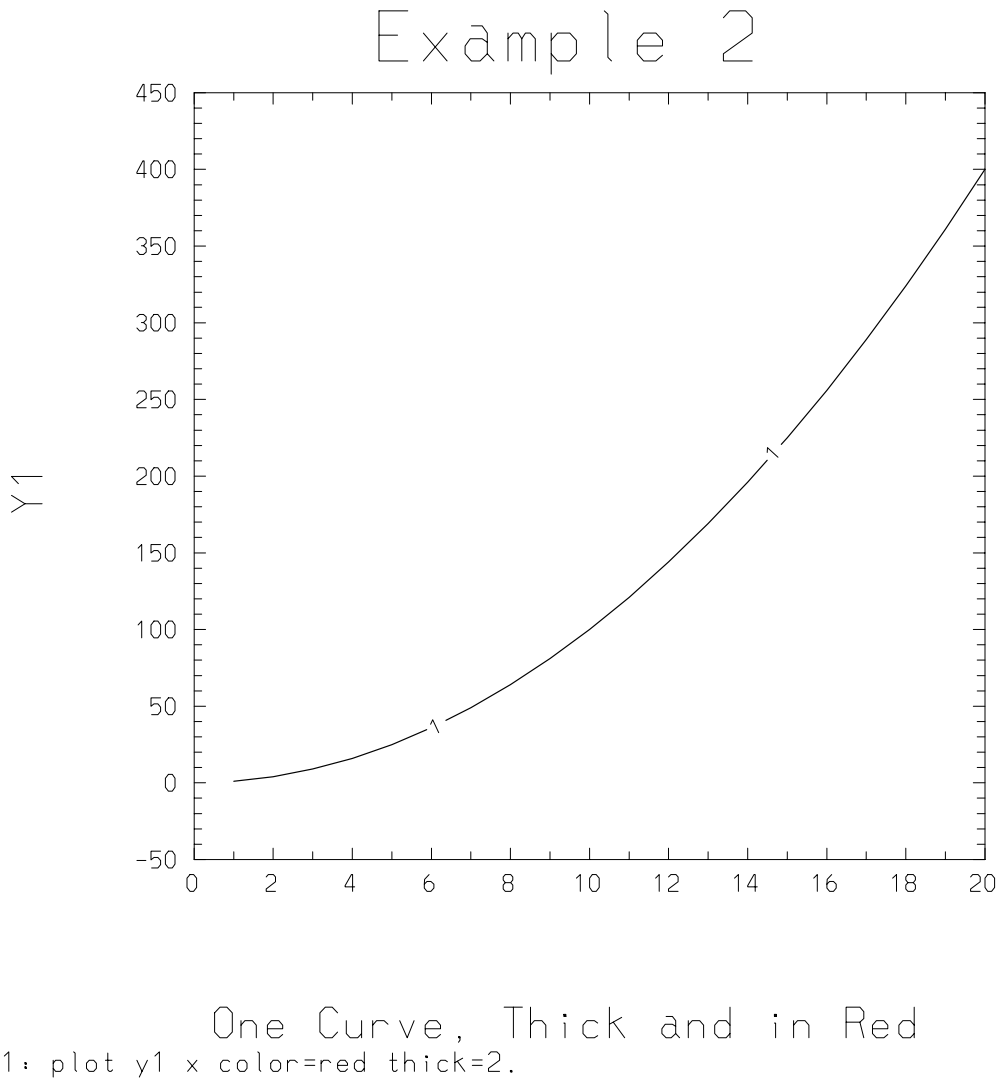


Figure 2.2: Example 2

```
# Example 3
nf
titles "Example 3","Three Curves on One Plot"
plot y1 x color=red
plot y2 x color=blue style=dashed
plot y3 x color=green style=dotted
sf
```

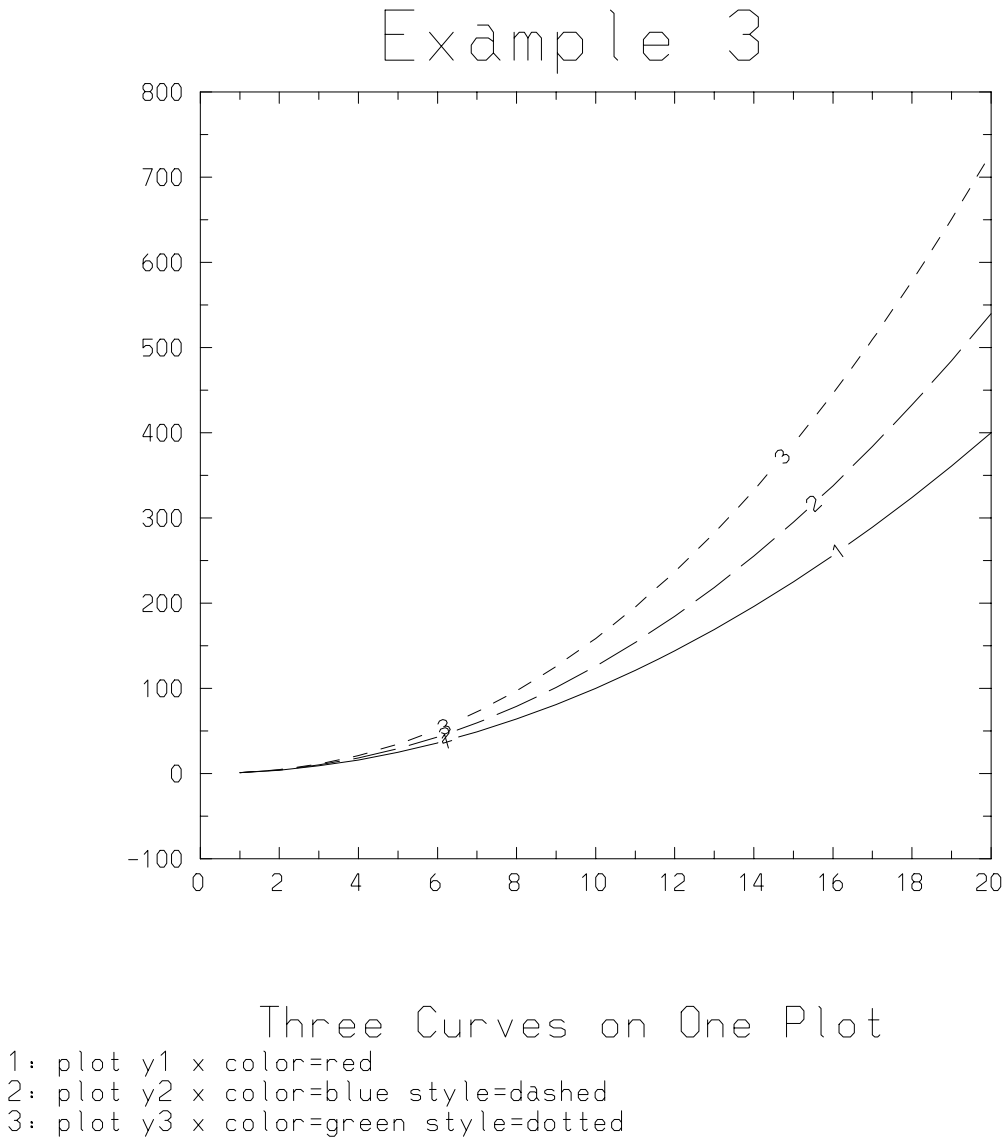


Figure 2.3: Example 3

```
# Example 4
nf; titles "Example 4",
   "Three Dashed Curves on One Plot, Frame Set, No Labels"
frame 5. 10. 10. 200.
attr labels=no style=dashed
plot y1 x color=red
plot y2 x color=blue
plot y3 x color=green
sf
```

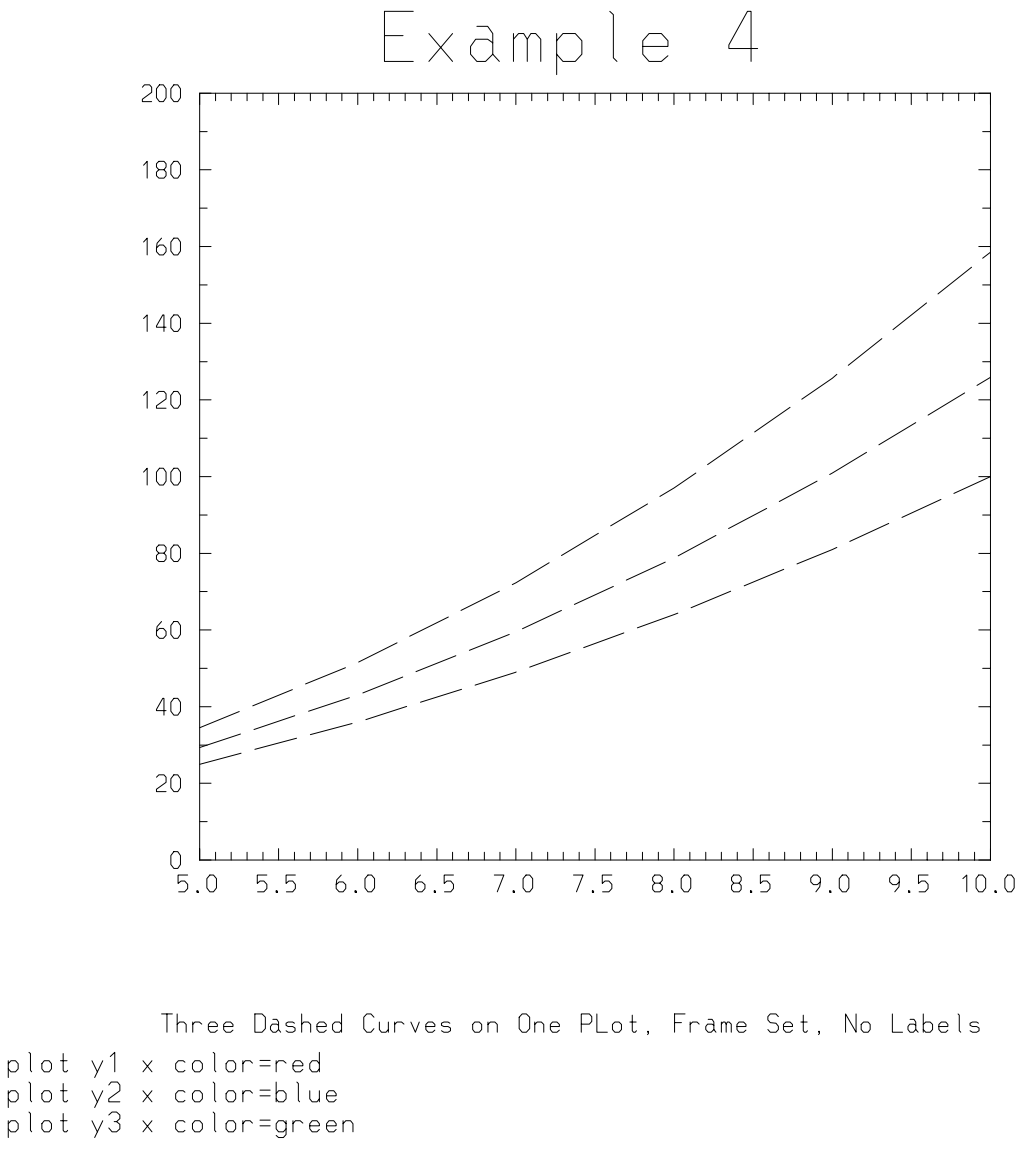


Figure 2.4: Example 4

```
# Example 5
nf
titles "Example 5","Three Curves on One Plot, Labeled"
# Note that we don't have to keep repeating x:
plot y1 x color=red labels="a"
plot y2   color=blue labels="b"
plot y3   color=green labels="c"
sf
```

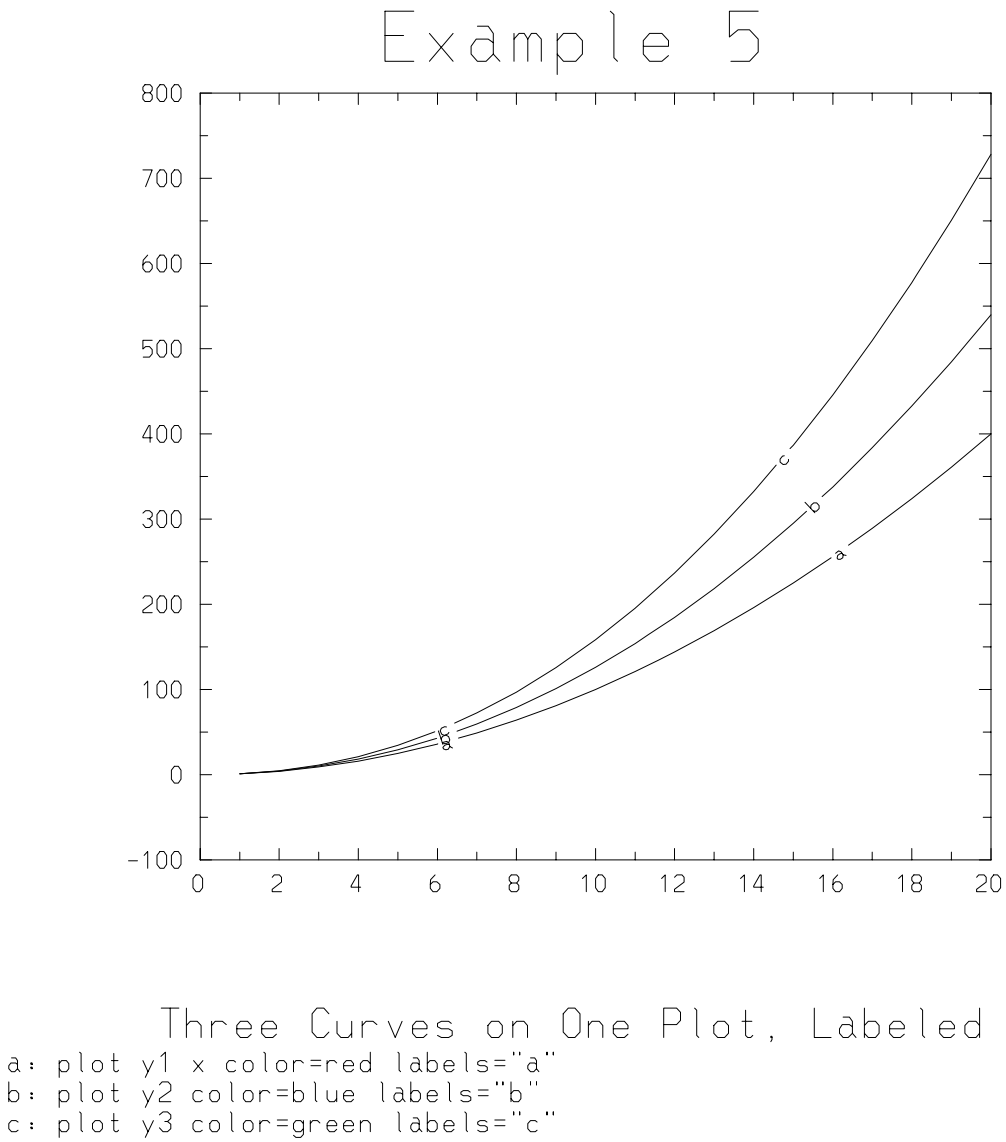


Figure 2.5: Example 5

```
# Example 6
nf
character*8 pwrs=["a","b","c"]
titles "Example 6",
   "Three Curves on One Plot, Labeled (Vector Syntax)"
plot [y1,y2,y3] x color=red labels=pwrs
sf
```

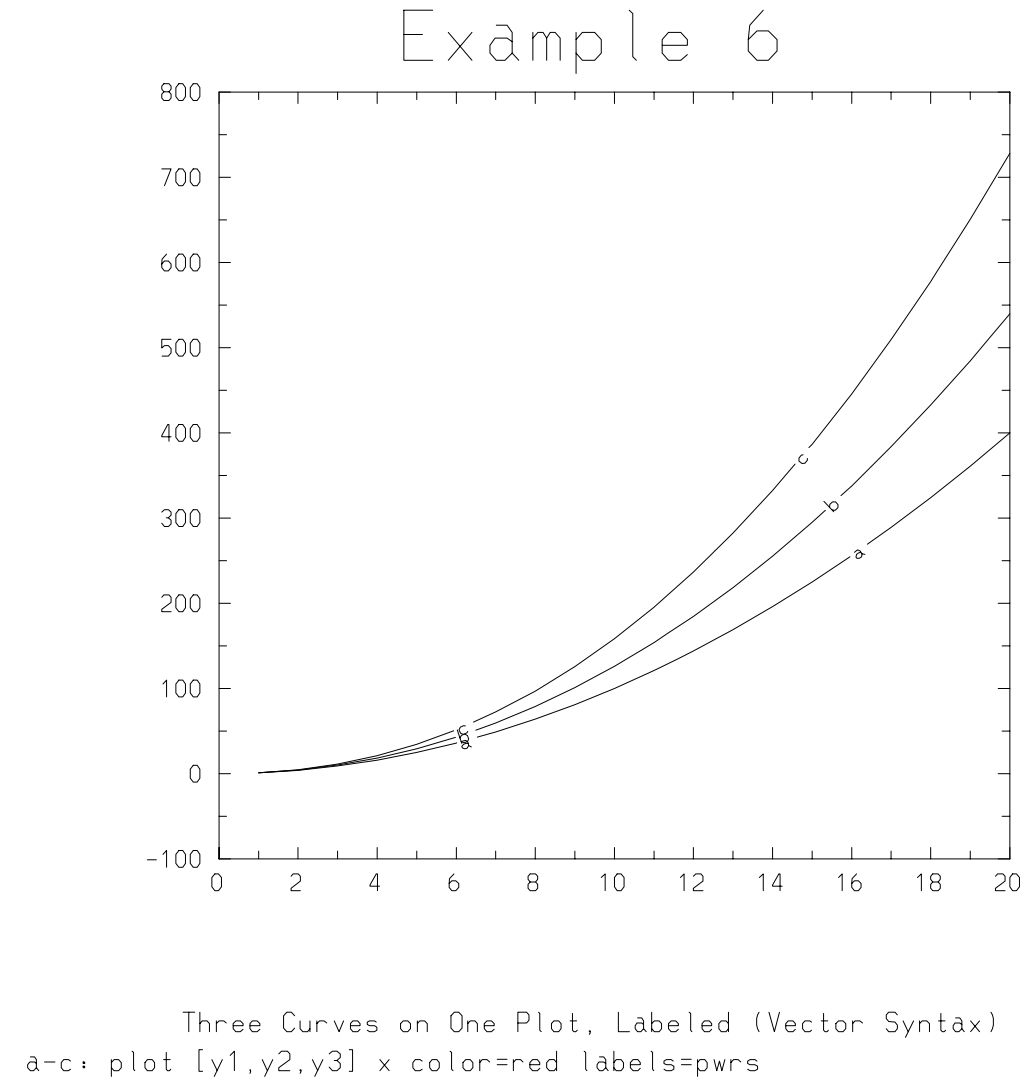


Figure 2.6: Example 6

```
# Example 7
nf
titles "Example 7","Three Curves on One Plot, Markers"
plot y1 x color=red mark=circle
plot y2 x color=blue mark=plus
plot y3 x color=green mark=cross marksize=2.
sf
```

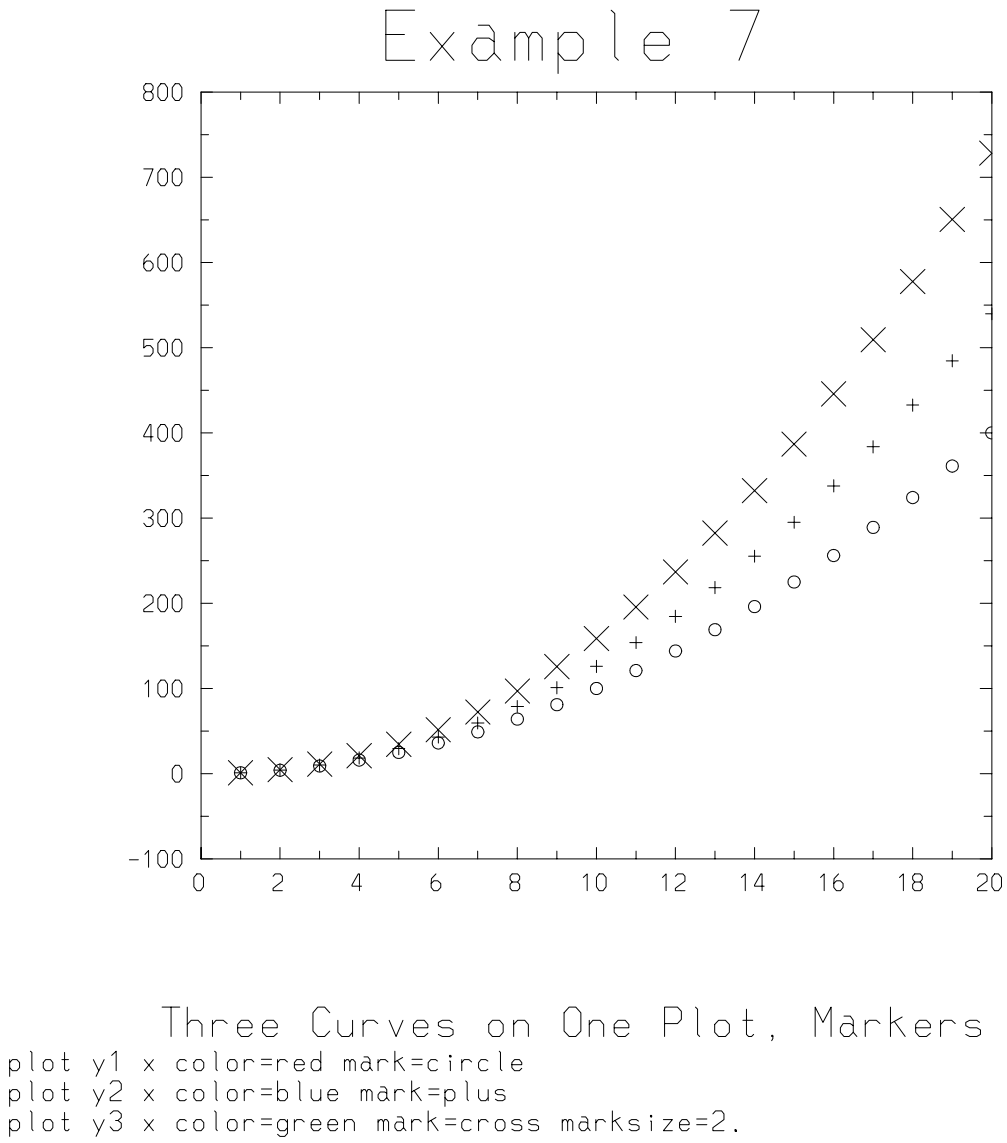


Figure 2.7: Example 7

```
# Example 8
nf
ezctitle="Supertitle appears on all subsequent frames"
titles "Example 8","Log(X)","Log(Y)"
plot [y1,y2,y3] x color=rainbow scale=loglog
sf
```

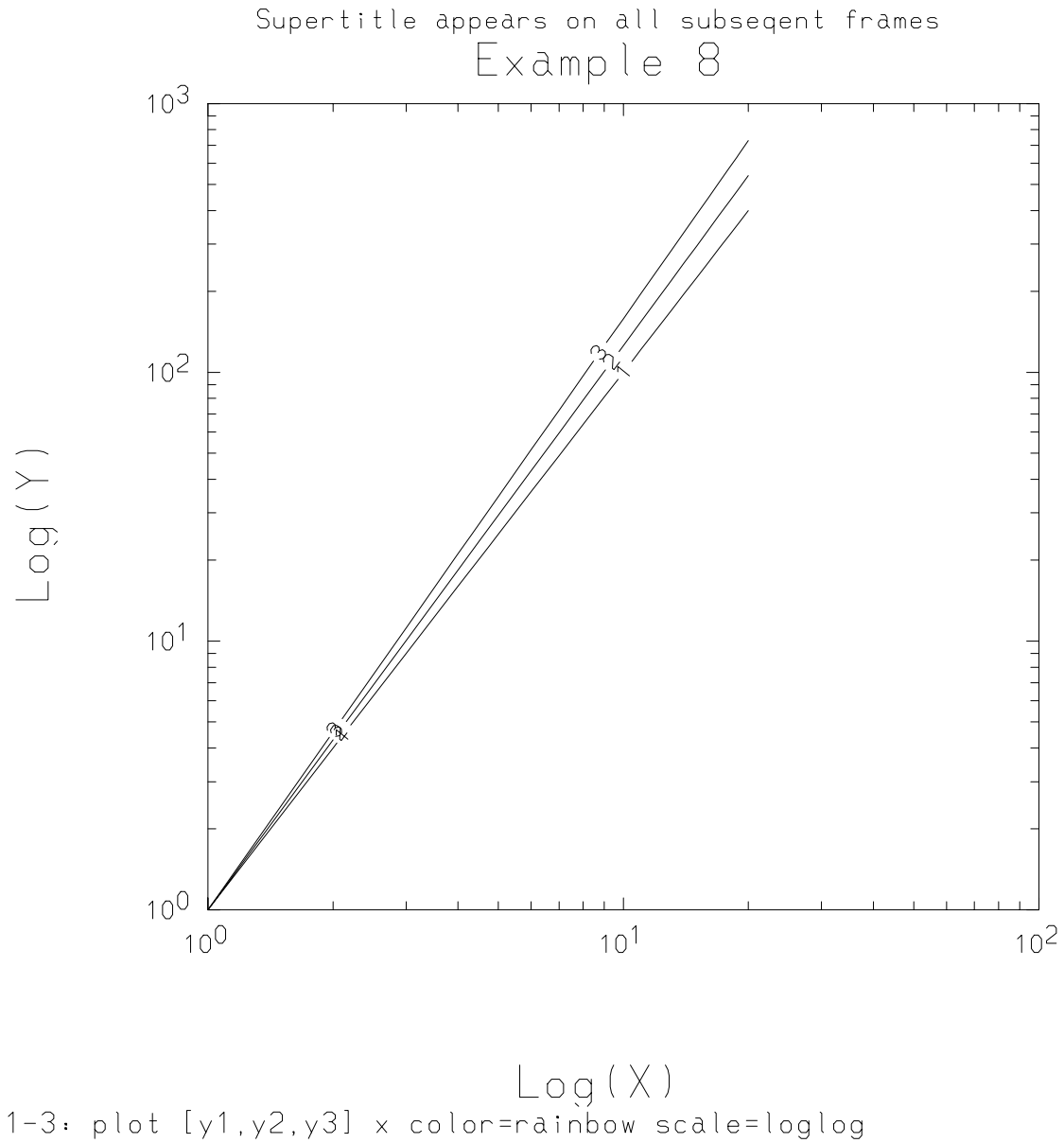


Figure 2.8: Example 8

```
# Example 9
nf
titles "Example 9","Contour Plot"
real x=iota(-5,5)
real y=x+6
real z=outer(x,y)
plotz z x y color=green lev=12
sf
end
```

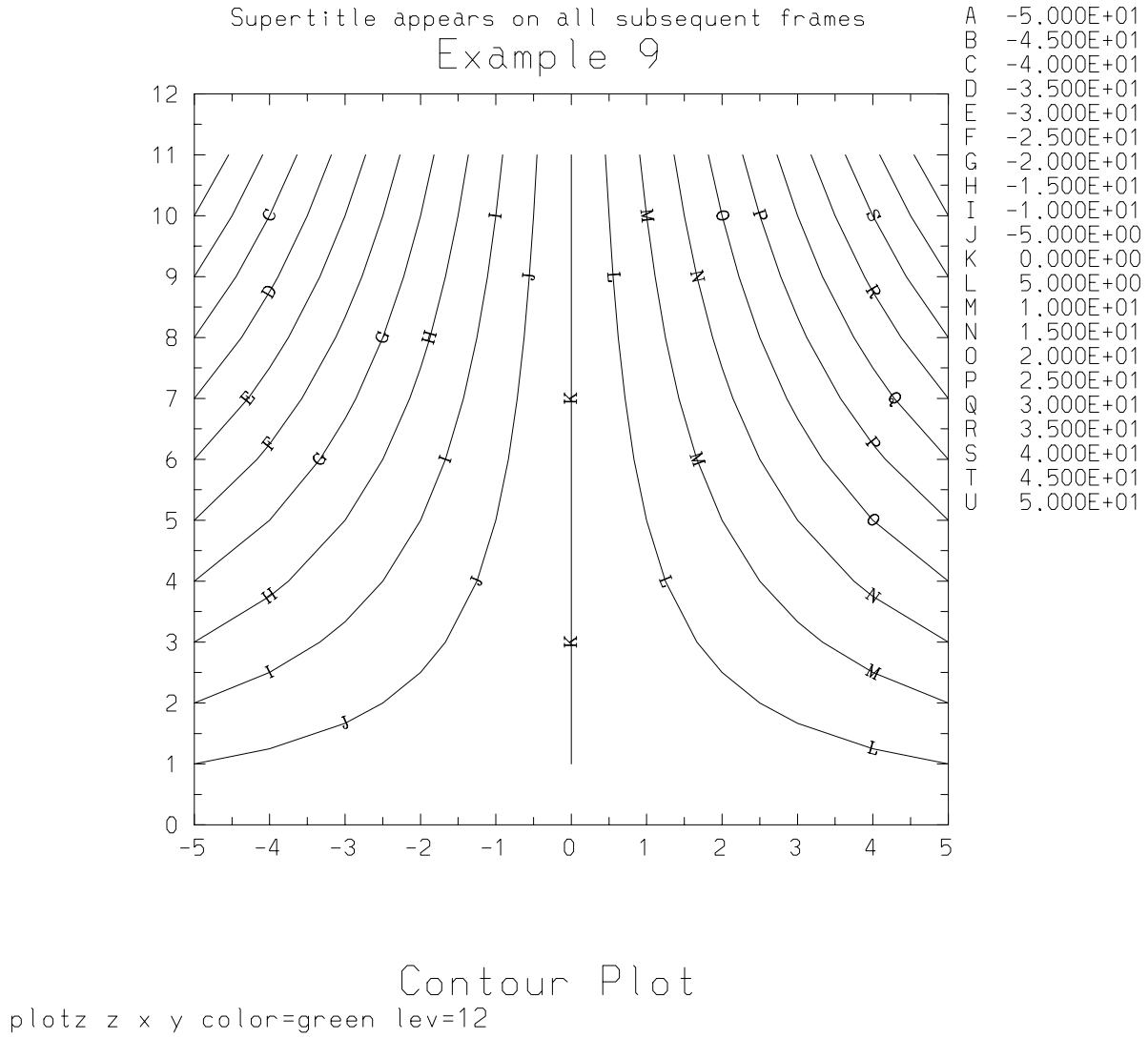


Figure 2.9: Example 9

2.2 Incorporating EZN in your program

Authors using the standard Basis makefile-creating program `mmm` will have EZN graphics by default. The `-nog` option to `mmm` will make the program without graphics. See the `mmm` man page for details. If you wish to make your own makefiles, read on.

Authors may add EZN to their program by including the file `ezn.pack`, found in the `include` subdirectory of the Basis installation (usually `/usr/local/basis/include`), in the input to the `config` program.

Load with the binary file `pkgezn.o` (found in the `lib` subdirectory of the Basis installation). You will also need to load with the appropriate NCAR and ATC libraries for your site. Basis comes with a utility `mmm` which can make the correct Makefiles for each site and architecture. If you wish to use your own Makefile system, the source for `mmm` can be used to deduce the appropriate information.

CHAPTER 3: Devices

The EZN package has commands to control graphics devices. The devices supported by EZN with ATC-GKS are CGM files, PostScript (PS) files, Xwindows, and Tektronix graphics terminals. The NCAR-GKS produces NCAR CGM files. With NCAR3.2 or later distribution, NCAR-GKS supports Xwindows. With NCAR4.0 or later, PS output is available and multiple windows are fully supported. The current version of the package (in Basis 11.8 or later) requires NCAR4.0.1 or later if NCAR-GKS is used.

A user can open multiple devices and direct the graphics output to different devices. For example, a user can open several Xwindows, even at different workstations, and display different frames in different windows for comparison. When the user is satisfied with the result of a certain frame, he/she can issue `cgm send` to record the frame into a CGM file, or ps `send` to record it to a PS file.

3.1 Device Commands

The device commands are of the form:

device-type device-command command-modifier

Here *device-type* can be: `cgm`, `ps`, `win`, `tek`. *device-command* can be: `on`, `off`, `close`, `send`, `list`, `slist`, or `colormap`. *command-modifier* can be: `mono`, `color`, or a string for *window-name*.

The device **`cgm`** is a CGM file. The CGM file stores the frames of graphics output. A CGM logfile, with suffix `.cgmlog`, is also created to record the frame numbers and each command issued in the frame. When ATC-GKS is used, a standard CGM file is produced, with suffix `.cgm`. When NCAR-GKS is used, the file produced, called an NCGM file, has a special format that NCAR utilities use. NCGM files have suffix `.ncgm` instead of `.cgm`. See 3.2 “CGM File Output” on page 17 for more details.

The device **`ps`** is a PostScript file, which has suffix `.ps`. The PS file stores the frames of graphics in the PostScript format. A PS logfile similar to the CGM logfile is created with suffix `.pslog`.

The device **`win`** (or **`tv`**) is an Xwindow on a certain “display”. The “display” is the network address set by the user’s environment variable `DISPLAY`. The variable `ezcdisp` can be set by the user to redirect the window display.

The device **`tek`** is a Tektronix Graphics Terminal (available with ATC-GKS only). *NOTICE*: Support for Tektronix output was frozen at the 11.3 release of Basis, and it may be withdrawn in a future release.

The command “`on`” opens a device if the device has not been opened. Then it activates the device. It has no effect on the device if it is currently active. “`open`” is a synonym for “`on`”.

The command "`off`" deactivates an opened device (but the linkage to the device for controlling still exists). The command "`close`" deactivates and then closes the device.

The command "`send`" sends the current frame to the specified device. If the target device is a CGM or PS file, the `send` command turns `on` the device (i.e. CGM or PS file), sends a frame, and then turns the file `off`. If the target device is an Xwindow or a Tektronix graphics terminal, then the current frame is re-sent to the device provided the target device is active.

The commands "`list`" and "`slist`" apply only to device **`win`**. For descriptions, see Section 3.3 "Working with Windows" on page 18.

For information about the command "`colormap`", see Section 3.5 "Setting the Colormap" on page 20.

The *command-modifier* is used to specify additional properties of the device. All devices default to `color`. When the user wants to override this default, he/she can supply the modifier `mono` when the device is opened the first time. (Modifier `color` is provided for compatibility with earlier versions in which `mono` was the default for device **`ps`**.)

The modifier *window-name* is used to label the window in order to identify it for future commands. The window name appears in the title bar of the window just opened. For more details, see Section 3.3 "Working with Windows" on page 18.

EZN keeps track of the number of active devices. If a plot command is issued without any active device, EZN will open a CGM file to accept the plot command.

Example 1

This example illustrates the use of the "`open`" and "`send`" commands.

```
win on
        # Open an Xwindow without name.
plot iota(20)**0.5
plot iota(20)**1.2
cgm send
        # Open CGM file, send a frame to it with two curves,
        # then immediately set CGM file to off.
nf
plot iota(15)**1.2
        # The plot appears on the window.
cgm send
        # Activate the CGM file to accept a frame,
        # then deactivate the CGM file.
```

```
ps send
        # Open PS file, send a frame to it,
        # then deactivate the PS file.
nf
plot iota(20)**1.5
        # The plot appears on the window.
ps send
        # Re-activate PS file, send a frame,
        # then deactivate the PS file.
end
        # Close all devices; close CGM file and PS file.
        # Examine logfiles to see what has been sent to each.
```

3.2 CGM File Output

Recall that the NCAR form of CGM file is different from the standard CGM file. Utilities for processing these files work on one format or the other.

The NCAR utilities `cgm2ncgm` and `ncgm2cgm` convert from one form to the other:

```
cgm2ncgm < foo.cgm > nfoo.ncgm
ncgm2cgm < nbar.ncgm > bar.cgm
```

The `gist` utility developed at LLNL can be used to view a standard CGM file interactively. You issue the command `gist` (or `agist`) to invoke the tool. Type "`gist -h`" for a short usage summary or see its man page for details.

The NCAR utilities `idt`, `ictrans`, and `ctrans` (found in `$NCARG_ROOT/bin`) can be used to view and print NCGM files. See their man pages for details.

The `idt` utility of NCAR can be used to view a NCGM file interactively. You issue the command `idt` to invoke the tool.

Here is how to print all the frames in an NCGM file on a monochrome laser printer:

```
ctrans -d ps.mono foo.ncgm | lpr
```

Here is how to view the frames in an NCGM file on a Tektronix 4010 terminal:

```
ctrans -d t4010 foo.ncgm
```

For a complete list of the devices supported by your local NCAR distribution, execute NCAR utility `gcaps` (found in `$NCARG_ROOT/bin`).

3.3 Working with Windows

In order to display graphics to an Xwindow it is necessary to first open it. By default, "`win on`" creates a window seven inches square. One may set variables `ezcwinht` and/or `ezcwinwd` to different sizes (in inches) before opening the window to override this default.

If multiple windows will be used, then window names must be provided on the `win` command to identify the windows for activation or closing. Only one window is active at any time for a given "display". The user can apply this feature to display different plots in different windows for comparison purposes.

Note that if a second window is opened, it will most likely appear on top if the first and will need to be repositioned to make both visible on the display. Study Example 2, below, for more information on working with multiple windows.

If there are only two windows connected, a "`win close`" to one of them automatically activates the other. On the other hand, if more than two windows are connected and the active window is closed, one must explicitly do a "`win on`" to activate the desired window.

To get a list of the currently open windows, type "`win list`". This will display at the terminal a list of the currently open windows in the order created, with their status `st` (T for active, F for inactive), workstation id `wsid`, connection id `wsconn`, and the associated display device. The exact format of the `wsconn` field is platform-dependent. For a short list containing only the first three columns type "`win slist`". Both forms are illustrated in the following example.

Example 2

Try out the following commands to gain experience working with multiple devices.

```
tv on window1 # (Same as "win on window1")
        # Create a window named "window1".
plot iota(20)**2 color=red
win on window2
        # Create another window named "window2";
        # the first window is deactivated.
plot iota(20)**1.5 color=green
        # The plot will appear on window2.  Note: both curves
        # appear, since EZN maintains a single display list.
nf
        # Set a new frame.
win list
        # List currently open windows.
```

```
 idx st  name              wsid     wsconn (hex)    display
 ------------------------------------------------------------
   1  F  window1          1001   240000100000000  128.115.36.49:0.0
   2  T  window2          1002   300000100000000  128.115.36.49:0.0
win on window1
       # Reactivate window1; window2 is now inactive.
win slist
       # Use short list to verify change of status.
 idx st  name
 ------------------------
   1  T  window1
   2  F  window2
plot iota(15)**2 color=blue
       # The plot will appear on window1, superimposed on what
       # was there already, because "nf" applies only to active
       # window.  Need a new "nf" after the "win on" to clear.
...
win close window1
       # Close window1; now window2 is automatically active.
nf
plot iota(10) color=magenta
win close window2
       # Close window2; now no devices are active.
nf
...
cgm on
       # Open a CGM file to accept the following plots.
       # (Note that this would happen automatically after the
       # next plot command, since no device is currently active.)
plot iota(20) color=yellow
       # A frame is send to the CGM file.
ps on
       # Open a PostScript file.
plot iota(20)**1.2
```

```
        # A frame has been send to both CGM and PS file.
nf
tek on
        # Open a Tektronix graphics terminal.
        # (CAUTION: This will produce very strange results
        # if you don't actually have one!)
plot iota(20)**1.8
        # A frame has been send to CGM, PS and Tektronix.
cgm close
        # Close CGM file.
ps close
        # Close PS file.
end
        # Implicitly close all active devices.
```

3.4 Setting the Background Color

The default background color is white for cgm and postscript devices, black for X-windows. The default can be overridden by executing one of the following calls *before* the device is opened:

`call ezcsetbw` -- set the background color to white.
`call ezcsetbb` -- set the background color to black.

The foreground color will be the complementary color.

3.5 Setting the Colormap

The "`colormap`" command can be used to alter the default colormap that is installed when a color device is opened.

The command has the form

device-type `colormap` *map-name*

where *map-name* should be "`idl`*n*", 1<=*n*<=16, to select one of the 16 IDL colormap that have been loaded into EZN. Some of these have alternate, more descriptive names:

Original	Alternate
`idl1`	`greyscale`
`idl2`	`bluescale`
`idl4`	`brownscale`

```
idl7        rainbow
idl8        pinkscale
idl9        greenscale
```

For complete control over the colormap, the user can specify *map-name* "`mycolormap`", which causes the RGB definition of the colormap to be loaded from the arrays `ezcred`, `ezcgreen`, `ezcblue`. The user should set these arrays to integer values in the range 0-256 before the device command.

The colormap setting must be done at the time a device is initialized. For device **`win`**, EZN will close the currently open window and create a new one with the requested colormap if only one window is open. For other device types, an error return will occur if a device of this type is already active. An error message will also result in case of multiple windows. You can use "`win list`" to display open windows and "`win close all`" to close them all. The new colormap will apply to the window opened with the next "`win on`" command.

Example 3

The following example sets up a colormap very much like the default.

```
# Create and install a colormap.
integer num = 249 # The size of the ezc color arrays.
integer num2=num/2
ezcred = 0.:256.:num
ezcgreen(1:num2) = 0.:256.:num2
ezcgreen(num2+1:num) = 256.:0.:(num-num2)
ezcblue = 256.:0.:num
cgm close  # Must open new CGM file to install new colormap.
cgm colormap mycolormap
```

CHAPTER 4: The EZN Graphics Model

4.1 The Additive Model

The basic model of this package is that of additive graphics commands to a single frame. That is, each graphics command adds objects (curves, meshplots, etc.) to a frame. The frame is not complete until a newframe "`nf`" command is issued. The user controls whether or not to see each step in building a frame or just viewing the completed frame by setting the variable `ezcshow` to `true` or `false`.

EZN begins in interactive mode (the variable `ezcshow` is true), so that each command that changes the frame causes the whole frame to be redrawn. However, Lasnex, and most other programs using EZN, will set `ezcshow` to false when making plots so that each frame is displayed only when finished. Lasnex users in particular should note that any "snapshot" plot will put EZN into this mode. If you stop the program and want to view the plots as they are made, you must either reset `ezcshow` to true or use the showframe "`sf`" command.

Caution: When using multiple windows in interactive mode, be aware that "`nf`" clears the display list, but only clears the currently open window. If you then change windows, you will have to issue another "`nf`" command to avoid overplotting any plot already on the window.

4.2 Controlling Layout

The standard EZN picture can be described as follows. The picture is divided vertically to allow a fraction (`ezccntfr`) of the right side of the picture to be used to list contour level values. The remaining left side is divided horizontally to allow a fraction (`ezclegfr`) at the bottom to be used for the legend. The remaining upper part of the left side consists of the axes and its labels surrounded by the four titles (whose relative size is controlled by `ezctitfr`). The supertitle `ezctitle` goes either at the top or bottom of the left part of the frame, depending on the value true or false of `ezcsuper`.

The function `ezcminsz(minsz)` can be used to set the minimum text size of numerical labels on the axes to a value between 6 and 24, inclusive. As this size increases, more room is left for the numerical labels. The default size is 12.

The space devoted to each of these components is, by default, allocated whether they are present or not. The variable `ezcfixed` can be set `false` to allow the space to be better used. This gives you as big a picture as possible. However, different pictures may allocate frame space differently, and hence no longer be directly comparable. For example, a mesh plot and a contour

plot will be different sizes because the latter will use some of the space for the contour legend.

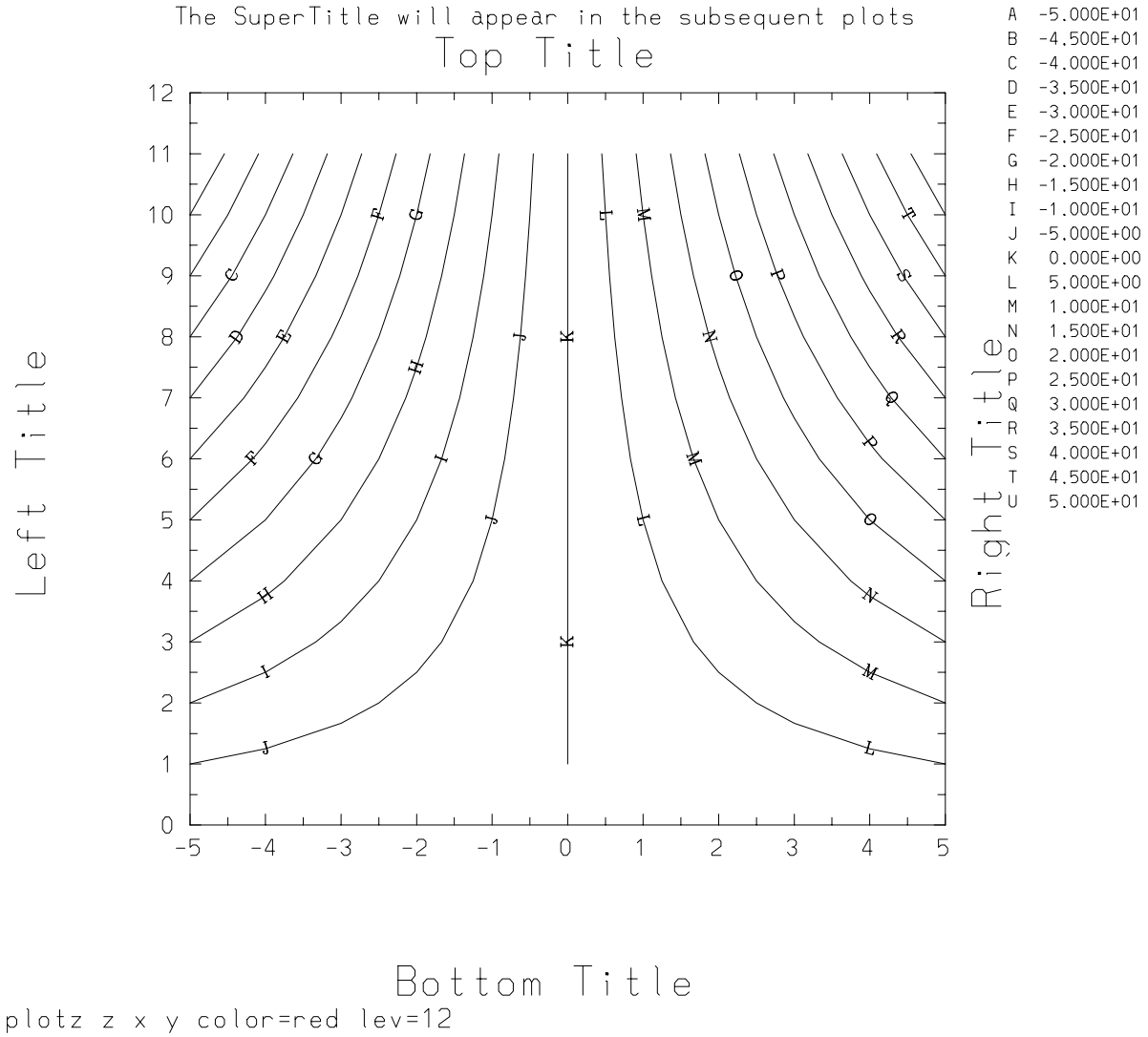


Figure 4.1: Example of Frame Layout

4.3 Plot Command Summary

Here is a summary of the commands which are described in the remainder of this manual.

- Attribute commands (CHAPTER 5: “Attributes”)

```
attr keyword=value, keyword=value
                # set color, thickness, etc.
```

- General plot commands (CHAPTER 6: “General Plot Commands”)

```
plot y x    # curves, markers
plotz z x y # contours
ploti cind # cell array plot
```

- Mesh-oriented commands (CHAPTER 7: “Mesh-Oriented Commands”)

```
plotm       # plot mesh
plotb       # plot mesh boundaries
plotc expr # plot contours of a mesh-based quantity
plotf expr # fillmesh plot
plotv       # plot velocity field
plotr lasernum # plot laser rays for Lasnex
```

- Polygonal-mesh commands (CHAPTER 8: “Polygonal-Mesh Commands”)

```
plotp x y   # plot polygonal mesh
plotpf expr x y # polygonal fillmesh plot
```

- Surface plot commands (CHAPTER 9: “Surface Plot Commands”)

```
srfplot     # wire-frame surface plot
isoplot     # wire-frame isosurface plot
```

- Frame control (CHAPTER 10: “Frame Control”)

```
frame xmin,xmax,ymin,ymax
nf          # begin new frame
sf          # display current frame
undo number # remove number’th command in the frame
```

- Text (CHAPTER 11: “Axes, Titles and Text”)

```
titles "top","bottom","left","right"
ezctitle = "supertitle for all frames"
text "message",x,y,charsize,angle,centering
ftext "message",x,y,charsize,angle,centering
stdplot << "X = " << x
output graphics# direct output to the graphics device
....ordinary Basis output
output tty      # direct output to the terminal
```

- Quadrant Control (CHAPTER 13: “Quadrant Mode”)

```
ezcquad(iquad)
ezcsquad(xmin,xmax,ymin,ymax)
```

- Interactive graphics (CHAPTER 14: “Interactive Graphics Tools”)

```
read interactive.in
zoom / unzoom     # zoom in on subframe
markp / markpp    # mark point(s)
markl / markll    # mark line(s)
marks / markss    # mark segment(s)
markr / markrr    # mark region(s)
markz / markzz    # mark zone(s)
```

You can use attributes and the values of user-settable variables to control the detailed behavior of these commands. Attributes are explained in the next chapter, variables in CHAPTER 15: “Control Variables and Defaults”.

CHAPTER 5: Attributes

A set of “attributes” such as color, line thickness, scale, marks, labels, etc., can be used to control the appearance of graphics objects or the layout of a frame.

5.1 Attribute Types

Some attributes affect the entire picture (such as scale, frame limits) while others affect the individual graphic objects in the picture (such as thickness, color).

If the attribute affects the entire picture, it will take effect immediately and we call it a *frame* attribute. If the attribute only affects the individual graphic object, we call it an *object* attribute. A special kind of object attribute (for mesh plots), which affects the current object and remains in effect until a frame advance or until another assignment is made to the attribute, is called "*sticky*". See “Attribute Table” on page 31 for a list of valid keywords, values and their attribute types.

The `grid` and `scale` attributes are examples of *frame* attributes. These attributes affect the entire picture. When these attributes are specified with the `attr` command or on an EZN graphic command line, a new picture is plotted with the grid and scale changed. (Note: This has the side effect of creating a new frame even if the variable `ezcshow` = `false`. To avoid the generation of extra frames, then, it is necessary to issue the `attr` and `frame` commands specifying the frame attributes *before* any plot commands for the frame.)

The `color` and `style` attributes are examples of *object* attributes. If these attributes are specified on a graphic command line, the color and line style are changed only for the objects generated by this command. If these attributes are specified with the `attr` command, only those objects added to the frame following the `attr` command will have these specified attributes. Some special attributes for the mesh plots such as `region`, `krange`, `lrange` are "*sticky*", i.e. the specifications of region, krange and/or lrange will affect the following mesh plots until the end of the frame or the values have been redefined.

If no attribute value is set explicitly by the user, a default value will be used for the attribute. These default values in turn can be changed by setting certain control variables. User specified default values will be in effect until new default values are assigned. For details, See CHAPTER 15: “Control Variables and Defaults” on page 101.

By specifying attributes and control variables, it is also possible to change many things about the layout of the picture, such as the portion of the picture used for the legend, the portion of the picture used for the contour level annotations, the size of the titles, and the minimum size of the text.

Usually all attributes will be re-initialized to their default values when a frame is advanced. However, setting the variable `ezcreset` to `false` will cause the attribute settings to last across

frames.

Examples

```
plot y1,x1
plot y2,x2
attr scale=linlog# Picture redisplayed.
nf
plot y1,x1
plot y2,x2
attr style=dashed# Only following curves affected;
       #        no redisplay yet.
plot y3,x3
plot y4,x4
nf
```

The attributes `legend`, `labels`, and `lev` can be either *frame* or *object* attributes. For example, `legend` can be set to `yes` or `no`, to indicate whether or not the graphic commands are to be listed at the bottom of the frame, thus supplying a handy index for each object generated by the graphic command on the frame. As an *object* attribute, `legend` can also be set to any arbitrary string for a particular command by specifying the `legend` attribute with the command, as in:

```
plot y x legend="Pressure versus density"
```

This results in the string "`Pressure versus density`" being listed as the command at the bottom, rather than "`plot y x`" which would result if this option were omitted.

The `legend` attribute used as an *object* attribute can also be used to suppress the legend for the current object, as in:

```
plot y x legend=" "
```

This causes the current command not be listed in the legend list.

Labels for the curves can be specified with the `labels` keyword. Labels must be quoted strings, or variables or expressions (including arrays) whose values are quoted strings. The attribute `labels` is also used to turn labelling on and off (by setting it to `yes` or `no`). When the attribute `labels` is used in this sense, it is a *frame* attribute. i.e., all existing and subsequent curves on the frame will be either labelled or not.

The attribute `lev` can be used to assign the number of contour levels or a vector of contour level values as an object attribute. When `lev=log`, it becomes a *frame* attribute, it sets the contour plots based on logarithmic scale.

5.2 attr: Setting Attributes

Calling Sequence

attr *keyword1=value1, keyword2=value2, ... , keywordN=valueN*

Description

The **attr** command assigns values to attributes. These keyword=value pairs can be either comma or space delimited. The value assigned to an attribute remains in effect until a frame advance is issued, or until another assignment is made to the attribute via the attr command (within the same frame). To make the values assigned to attributes remain in effect across frame advances, set variable ezcreset to false.

To make a permanent change to a default, change the corresponding variable. For a list of these, See CHAPTER 15: “Control Variables and Defaults” on page 101.

Examples

In the first example, the scale is set to loglog, the line style is set to dashed. Since the default value for variable ezcreset was used, the attributes set only remained in effect until the next frame advance. After that, the attributes are reset to their default values.

```
# ezcreset=true (default)
# Settings remain in effect only until next frame advance.
attr scale=loglog,style=dashed
plot y1,x1
plot y2,x2
nf
plot y3,x3# scale,style reset to defaults.
```

In the second example, variable ezcreset is set to false. This time the attr command remains in effect across frame advances. Hence, the line thickness remains set to 1.2 across frame advances.

```
ezcreset=false
# Settings remain in effect across frame advances.
attr thick=1.2
plot y1,x1
plot y2,x2
nf
plot y3,x3# Thickness still 1.2.
```

Or, we could accomplish the same thing more simply by making a permanent change to the default thickness:

```
# ezcreset=true (default)
defthick=1.2
plot y1,x1
plot y2,x2
nf
plot y3,x3# Thickness still 1.2.
```

5.3 Attribute Table

The following is an alphabetical list of all allowable attribute keywords. Refer to individual plot commands for more specific information.

TABLE 1. **attr: Attribute Table**

Keyword	Type	Value	Description
arrow	object	no	No arrows on curve (default).
		yes	Plot arrows on curve.
bnd	object	no	Plot full mesh (default).
		yes	Plot region boundaries only.
color	object	bgcolor fgcolor	The default background / foreground color used by EZN.
		color	Use one of the following 16 named colors (default=fgcolor): red, green, blue, cyan, magenta, yellow, coral, yellowgreen, springgreen, slateblue, skyblue, orangered, gray33, lavender, orchid, gray70.
		rainbow	Colors run down through the list of named colors. (*See* note after table for more details.)
		filled	Color fill the contour band, ranging from blue to red.
		rfill	Color fill the contour band, ranging from red to blue.
		fillnl	"filled" without contour lines.
		rfillnl	"rfill" without contour lines.
		power	Ray color varies along path to show intensity. (*See* 7.5 “plotr: Lasnex Rayplots”.)
		relpow	Ray color shows relative intensity. (*See* 7.5 “plotr: Lasnex Rayplots”.)
cscale	object	lin	Use linear color mapping (default). (*See* 7.3 “plotf: Fillmesh Plot”.)
		log	Use logarithmic color mapping. (*See* 7.3 “plotf: Fillmesh Plot”.)
		normal	Use color mapping based on the normal distribution. (*See* 7.3 “plotf: Fillmesh Plot”.)

TABLE 1. **attr: Attribute Table (Continued)**

Keyword	Type	Value	Description
grid	frame	no	No reference grid
		tickonly	Tick marks only (default)
		x	x rulings
		y	y rulings
		xy	x and y rulings
kcolor	object	*color*	Use *color* for k-lines. (*See* color; default: current color attribute)
krange	sticky	*kmin:kmax:kinc*	Range for k-lines in mesh plot. (default=1:kmax:1)
kstyle	object	none	No lines in k direction.
		style	Use *style* for k-lines. (*See* style; default=solid)
labels	frame	yes	Curves/marks are labelled in the order added (default).
		no	No labels displayed.
	object	*str*	Label next curve with *str.* *str* can be a vector for multiple curves.
lcolor	object	*color*	Use *color* for l-lines. (*See* color; default: current color attribute)
legend	object	*str*	User-specified legend in quotes (maximum 120 characters). By default, the command line is used.
	frame	yes	Legend plotted below the frame (default).
		no	No legend plotted.
lev	object	*ival*	Number of levels (default=8). (For complete specification, *see 6.2.1 "Contour Levels".)*
		[*rval*]	Vector of contour levels.
	frame	linear	Linear contours (default).
		log	Logarithmic contours.
lrange	sticky	*lmin:lmax:linc*	Range for l-lines in mesh plot. (default=1:lmax:1)
lstyle	object	none	No lines in l direction.

TABLE 1. **attr: Attribute Table (Continued)**

Keyword	Type	Value	Description
		style	Use *style* for l-lines. (*See* style; default=solid)
mark	object	asterisk	Use asterisk marker.
		circle	Use circle marker.
		cross	Use cross marker.
		dot	Use dot marker.
		plus	Use plus marker.
		x	Use x marker.
marksize	object	*rval*	Scaling factor for markers (default=1.).
point	object	no	Physics quantity is zone-centered (default). (*See* CHAPTER 7: "Mesh-Oriented Commands".)
		yes	Physics quantity is defined at mesh points.
region	sticky	all	Display all regions in mesh plots (default).
		[*ival*]	Vector of desired region numbers.
rsquared	object	*rval*	Multiquadric r-squared parameter.
		0.	Program calculates (default).
scale	frame	linlin	Both x and y axes linear (default)
		linlog	x-axis linear, y-axis logarithmic
		loglin	x-axis logarithmic, y-axis linear
		loglog	Both x and y axes logarithmic
		equal	Both x and y axes linear, scales equalized
style	object	solid	Solid lines (default)
		dashed	Dashed lines
		dotted	Dotted lines
		dotdash	Dot-dashed lines
		ltor	Mark curve with arrows pointing left-to-right.
		rtol	Mark curve with arrows pointing right-to-left.
		pm	Plus/minus (for contour plot)

TABLE 1. **attr: Attribute Table (Continued)**

Keyword	Type	Value	Description
		none	Background color (invisible) lines
thick	object	*rval*	Line thickness multiplier (default=1.)
vsc	object	*rval*	Vector scaling factor (default=0.05).
zlim	object	[*zmin*,*zmax*]	Limits to be used in defining color-map for fillmesh plot (default=limits of plotted array).

Note: In the `color` array that assigns names to colors, "bgcolor" has index 0, "fgcolor" index 1. The named colors are numbered from 2 to 17. "color=rainbow" cycles through indices 1 through 13 only. When applied to a vector of curves, the cycle starts at 2 (red); for contour lines, the cycle starts at 4 (blue).

CHAPTER 6: General Plot Commands

This chapter describes the EZN general-purpose plot commands.

6.1 plot: Plotting Curves and Markers

Calling Sequence

`plot` *yexpr,xexpr,<keylist>*

Description

The **`plot`** command plots line segments connecting points or discrete markers at the points. Markers are plotted at the data points, without connecting line segments, when the attribute `mark` is set to one of the valid marker types. (An invalid marker type is treated as `mark=x`.)

The default scaling factor for markers is 1.0, the default line style is `solid`, and the default line thickness is 1.0. To override these values, set the attributes `marksize`, `style`, or `thick`, respectively. (Due to the Xwindows "clip" characteristics of NCAR, markers that would be partially drawn beyond the frame limits are completely clipped. In order to see the markers at the frame limits, the user needs to use a `frame` command to extend the limits to include the whole markers.)

By default, the curve is plotted with both axes on a linear scale. For logarithmic plots, set attribute `scale` to "`linlog`", "`loglin`" or "`loglog`".

If neither *yexpr* nor *xexpr* is specified, the current picture is redisplayed. Otherwise, *yexpr* is an array of y-axis values, *xexpr* is an array of x-axis values, and <*keylist*> is a list of optional attributes specified by pairs of keywords and values.

If *xexpr* is not specified, then *yexpr* is plotted against the previous version of *xexpr*. If this value does not exist, then *yexpr* is plotted against the index of *yexpr*. If plotting against the index of *yexpr*, the lower and upper subscripts of *yexpr* are used as the starting and ending points, respectively.

If *yexpr* differs in length by one from the length of *xexpr*, whether explicitly or implicitly specified, the longer of the two may be automatically averaged to shorten it. Set variable `ezcnocx` or `ezcnocy` to `false` to disable averaging. If averaging is not permitted, the command is an error and no object is added to the frame.

If the arguments are two-dimensional arrays, `plot` plots the corresponding columns of *yexpr* and *xexpr* to produce multiple curves at once. Multi-dimensional arguments are reduced to two-

dimensional by collapsing any higher dimensions. If *xexpr* is one-dimensional, then each column of *yexpr* is plotted against it.

Optional Attributes

The following optional attributes can be specified with this command. For object attributes, they are local to the command specified; i.e., they are not remembered across commands.

`grid`, `scale`, `style`, `thick`, `color`, `arrow`, `labels`, `font`, `mark`, `marksize`, `legend`

If optional attributes are given on the plot command line, they are specified in the usual form:

`key1=value1,key2=value2,...,keyN=valueN`

To set an object attribute across commands use the **`attr`** command. See “Attribute Table” on page 31 for descriptions of the values which can be assigned to these keywords.

Although it is recognized, the `font` attribute currently has no effect.

Examples

In this example, three curves will be superimposed. The first plot command will plot a curve with dashed lines, the second plot command will mark circles twice the default size, and the third plot command will plot the curve in red. Since the first plot command does not specify *xexpr*, y will be plotted against an array spanning from 5 to 15. In the second and third plot commands, the y values are plotted against the previous expression of x. The curves are labelled 1 and 3 respectively. (“Marked” plots are not labelled.)

```
real y=iota(5,15)
plot y,style=dashed                # plot curve
plot y+1,mark=circle,marksize=2    # plot markers
plot y+2,color=red                 # plot curve in red
nf
```

If you enter the above commands at the terminal, you will see three frames displayed in turn as the graphic objects are built up, and the `nf` command will clear the screen. If you repeat the experiment with `ezcshow=false`, you will not see any graphic objects at all until the `nf` command, at which point the completed frame will appear.

The next example replots two curves with an xy-grid added.

```
real x=0.5*iota(1,10)
real y=x**2
plot y,x
plot y-1,x
plot grid=xy
nf
```

In the next example, the first plot command will plot three curves, y,y+1,y+2 against the same x, labelled “a”, “b”, and “c”, respectively. The second plot command will plot two curves, y+3 against x+1 and y+4 against x+2, labelled 4 and 5, respectively. (Note that the curve number continues to increment even if the number is not the curve label.)

```
plot [y,y+1,y+2],x,labels=["a","b","c"]
plot [y+3,y+4],[x+1,x+2]
nf
```

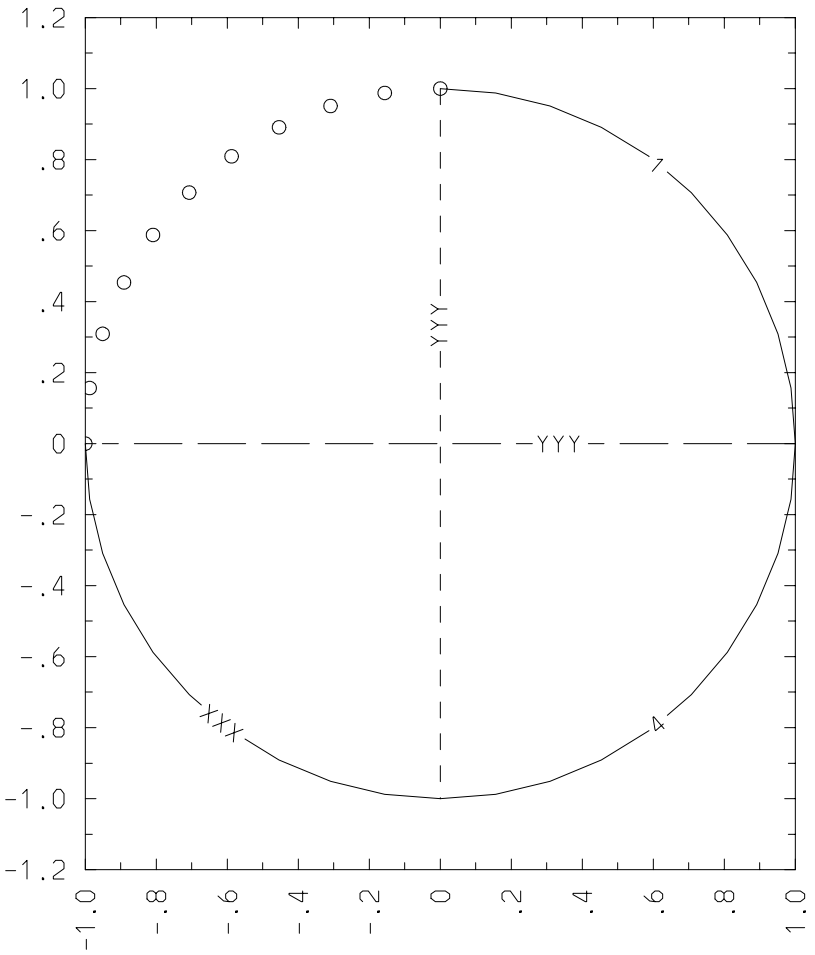
The fourth example shows how to set the legend.

```
plot y,x,legend="this is my legend"
nf
```

The fifth set of examples graphs the unit circle and x and y axes in a variety of styles and also illustrates how the `labels` attribute works. Comments in the code explain what happens on the frame.

```
attr scale=equal
# Set x and y scales be equal:
$a=(pi/2.)*iota(0,10)/10.
# Curve in first quadrant labelled with 1:
plot cos($a) sin($a) legend = "first quadrant"
# Curve in second quadrant not labelled "Q2" since
#   drawn with a "mark":
plot cos($a) -sin($a) labels ="Q2" mark=circle
# Third quadrant drawn and all labels turned off, but
#   label "XXX" is still associated with quadrant 3:
plot -cos($a) -sin($a) labels=no labels="XXX"
# All labels turned back on, including "XXX" in quadrant 3;
#   quadrant 4 labelled with 4:
plot -cos($a) sin($a) labels=yes
attr labels="YYY"
# The following two curves will now be labelled with "YYY":
plot 0*ones(11) 1.//((5.-iota(10))/5.) style = dashed
plot 1.//((5.-iota(10))/5.) 0*ones(11) style = dotted
```

See the following figure for the completed frame. (Note that the figure was generated before NCAR graphics totally clipped markers that are partially beyond the frame limits, so the picture you get by executing these commands will differ somewhat from what you see here.)




```
1: first quadrant
plot cos($a) -sin($a) labels="Q2" mark=circle
XXX: plot -cos($a) -sin($a) labels=no labels="XXX"
4: plot -cos($a) sin($a) labels=yes
YYY: plot 0*ones(11) 1.//((5.-iota(10))/5.) style=dashed
YYY: plot 1.//((5.-iota(10))/5.) 0*ones(11) style=dotted
```

Figure 6.1: Example of Labelling and Legend Specification

6.2 plotz: Plotting Contours

Calling Sequence

`plotz` *fexpr,xexpr,yexpr,<keylist>*

Description

The **`plotz`** command plots contours of a surface defined by *fexpr* above the point set described by *xexpr* and *yexpr*. *<keylist>* is a list of optional keywords and values.

There are three allowed types of data for contour plots:

- Gridded data: *xexpr* and *yexpr* are one-dimensional arrays, say x and y, and *fexpr* is a two-dimensional array, say z, such that z(i,j)=f(x(i),y(j)), i=1,...,length(x), j=1,...,length(y). In order for *xexpr* and *yexpr* to form a valid rectangular grid, each array must contain either strictly increasing or strictly decreasing values.
- Mesh data: *fexpr*, *xexpr* and *yexpr* are all two-dimensional arrays of the same shape. In this case, *xexpr* and *yexpr* form a logically rectangular mesh and *fexpr*(i,j) is the value associated with point (*xexpr*(i,j),*yexpr*(i,j)). For mesh-based data, a plot of this type can also be generated by the `plotc` command; see Section 7.2 "plotc: Plotting Contours" on page 54.
- Scattered data: *fexpr*, *xexpr* and *yexpr* are all one-dimensional arrays of the same length. In this case, a rectangular mesh containing the data is created and *fexpr* is interpolated to this mesh by the MultiQuadric (MQ) method. This is the only case in which the optional attribute `rsquared` is used.

Note: *fexpr* can also be the name of a function or macro which, when called with no arguments, returns an array of values of the appropriate shape.

Optional Attributes

The following optional attributes can be specified with this command. For object attributes, they are local to the command specified; i.e., they are not remembered across commands.

`grid`, `scale`, `thick`, `style`, `font`, `mark`, `marksize`, `lev`, `color`, `rsquared`, `legend`

If optional attributes are given on the plot command line, they are specified in the usual form:

`key1=value1,key2=value2,...,keyN=valueN`

To set an object attribute across commands use the **`attr`** command. See “Attribute Table” on page 31 for descriptions of the values which can be assigned to these keywords.

The default line style is `solid` and the default line thickness is 1.0. The default color is the foreground color. To override these defaults, set attributes `style`, `thick`, `color`, respectively. The `mark` attribute will cause markers to be plotted at each of the mesh points, in the foreground color.

Although it is recognized, the `font` attribute currently has no effect.

6.2.1 Contour Levels

Contour levels are controlled by the `lev` attribute. The attribute `lev` can be used to specify the levels of contours, the scale of the contours (*linear* or *logarithmic*), or a list of specific values for the contour levels. The attribute `lev` can be set either on a plot command or with an attribute command such as "`attr lev=foo`". Like any such attribute, if set with `attr` it applies to all `plotz` commands on that frame, except those that override it with a "`lev=` " of their own. However, if a vector of values is specified for `lev`, it will be lost at the next frame advance. There is currently no way to specify such a list to be used on all frames.

In "`lev=foo`", `foo` can be:

- `linear`: at least abs(`deflev`) linear levels;
- `log`: abs(`deflev`) logarithmic levels;
- *n>0*: at least *n* linear levels;
- *n<0*: abs(*n*) logarithmic levels;
- a real or double precision list of values.

The contour plot is generated by the NCAR Conpack package. When NCAR chooses linear levels, it chooses at least the number specified but may choose up to 2**n* levels.

The default value of `lev` is in the variable `deflev`, whose value is 8; hence, the default is at least 8 linearly-spaced contour levels.

Every contour line is labeled. The variable `ezclabel` can be set to "`alpha`", "`on`", or "`off`". This will result in contours which are labeled with single letters, with contour level values, or with nothing, respectively. The default is `ezclabel=alpha`.

The special "`style=pm`" for a contour plot invokes the mode where positive contour lines are plotted using solid lines, and negative contour lines are dashed. (Mnemonic: `pm` means `plus/minus`.)

6.2.2 Contour Control Parameters

After a contour plot has been displayed, a set of variables is available to review the information about the set of contour levels used by NCAR. Do "`list Contours`" to see this list. Do "`list Random_Contour_Plots`" for variables related to the MultiQuadric interpolated values in the scattered data case.

For more detailed control of NCAR Conpack, the user may use three routines `cpsetr`, `cpseti`, and `cpsetc` to set real, integer and character parameters respectively. The following parameters in Conpack are set by EZN; the user should not change them:

```
CLS - contour level selection
NCL - number of contour levels
CLV - contour level
CLU - contour level usage
CLD - contour level dash pattern
CLL - contour level width
```

```
LLT - level label text
SET - calling of set by Conpack
SPV - special value
ORV - out of range value
ILT - info level text
SFS - scale factor selector
```

Conpack parameters whose values EZN sets just once during its initialization are:

`CWM - character width multiplier`

The default is 1.2. Set this bigger or smaller to control the size of the contour labels.

`HLT - hi/lo text labels`

The default is `"H'L"`. Set to blank via command `cpsetc("HLT"," ")` to remove the `H` and `L` labels.

`PC1 - real contour label positioning parameter`

Extremely short contours are not labelled. This parameter governs how short a contour can have a label.

Conpack parameters that the user might experiment with are:

`DPV` - This integer controls the distance between the labels on a contour line.

The default corresponds to 3.

`LLP` - This controls the contour label positioning.

The inquiry routines `cpgeti`, `cpgetr` and `cpgetc` may used to find out the current setting of those parameters. The user should refer to the NCAR document "CONPACK, A Contouring Package" (available on the web at http://ngwww.ucar.edu/ngdoc/ng/supplements/conpack/) for detailed information.

6.2.3 Contour Color Fill

The `color` attribute for a contour plot can be used to generate color filled contour bands. This is an application of the "area" concept of NCAR. Each contour band is a closed polygon (coupled with frame boundaries if necessary) which can be filled with color. The user can set `color=filled` to fill the contour levels with colors ranging from *blue* to *red* with *increasing* altitude. Setting `color=rfill` will fill with colors ranging from *red* to *blue*. When color fill is applied, the contour lines may become unnecessary. The user may specify `color=fillnl` or `color=rfillnl` to avoid the contour lines being drawn.

For the advanced user, a different range of colors can be assigned when the default colormap is changed. See "Setting the Colormap" on page 20 for more information.

6.2.4 Contour Level Annotations

For the contour plots, the contour level annotations can be shown in the right margin of the frame

under user's control. The value of control variable `ezccntfr` is the fraction of the whole frame on the right allocated to display this information. It lists the labels and their corresponding contour level values. (If labelling is suppressed, `ezclabel=off`, then only the contour level values will be in the annotation.)

The variable `ezcconkey` is used to control the appearance of the contour level annotation. Setting `ezcconkey=off` will cause the frame not to display the contour level annotation. (However the portion of the frame for contour level annotation is still allocated. To utilize the whole frame without contour level annotation, set `ezcfixed=false` and `ezccntfr=0.`) The default is `ezcconkey=on`.

The contour level annotation is color coded for easy association with the contour lines. The color assigned is the color of the contour level. There are several control variables that may be used to customize this contour level annotation. The variable `ezccksfill` specifies either *solid* color fill or *hollow* fill (i.e. just color the border) for each cell containing the numerical annotation. `ezccksfill="solid"` specifies the solid fill; any other value, e.g. `ezccksfill="hollow"` (the default), will make a hollow fill. In a rare case, there may be two contour plots in the same frame. If conflicting colors are assigned to the same contour level, the contour level annotation will be hollow filled with a white border.

Example

The following example plots a matrix z versus vectors x and y. (Repeated from Example 9 in CHAPTER 2: “Introduction to EZN”.)

```
real x=iota(-5,5)
real y=x+6
real z=outer(x,y)
plotz z x y color=green lev=12
nf
```

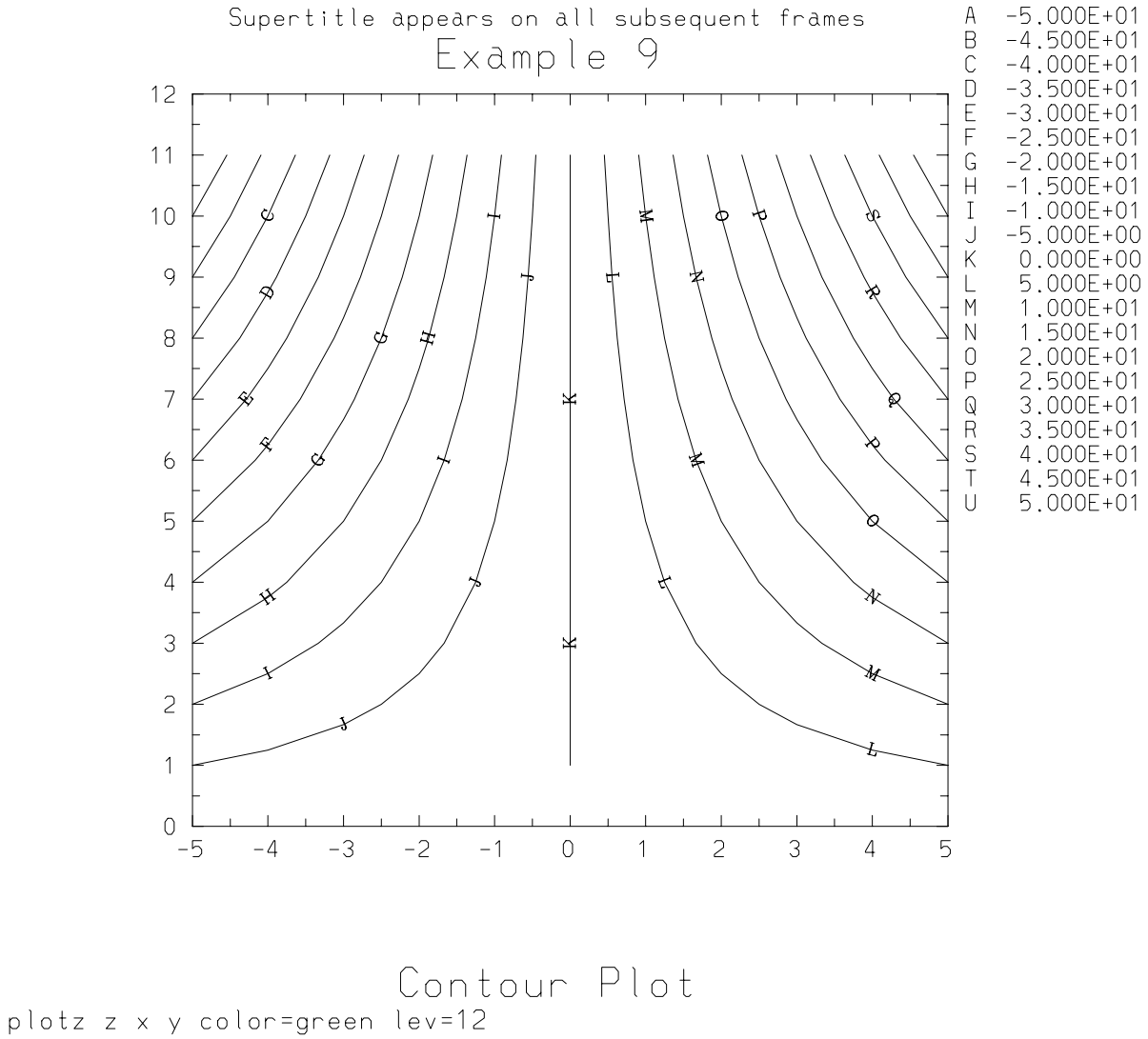


Figure 6.2: Example of Contour Plot with Level Annotation

6.3 ploti: Cell Array Plots

Calling Sequence

`ploti` *cell-indices*`[,`*xmin*`,`*xmax*`,`*ymin*`,`*ymax*`],<`*keylist*`>`

Description

The **`ploti`** command is used to plot cell arrays in Basis. This is an application of the "area" concept of NCAR. The argument *cell-indices* is a two-dimensional array of color cell indices, which can be generated using the vector-to-color conversion functions described below. <*keylist*> is a list of optional keywords and values.

The user translates a two-dimensional array of physics quantities to a two-dimensional array of color indices, and cell array plot displays the corresponding colors in a rectangular matrix of color cells. The optional arguments *xmin*,*xmax*,*ymin*,*ymax* specify actual limits for the physics data, in order to display correct *x*,*y*-labels on the plot axes. If no frame limits are given, `ploti` will use the square [0.,1.]x[0.,1.] to plot the color array.

For mesh-based data, a more realistic display may be obtained by using the `plotf` command instead; see Section 7.3 "plotf: Fillmesh Plot" on page 57.

Optional Attributes

The following optional attributes can be specified with this command. For object attributes, they are local to the command specified; i.e., they are not remembered across commands.

`grid`, `scale`, `font`, `legend`

If optional attributes are given on the plot command line, they are specified in the usual form:

`key1=value1,key2=value2,...,keyN=valueN`

To set an object attribute across commands use the **`attr`** command. See "Attribute Table" on page 31 for descriptions of the values which can be assigned to these keywords.

Although it is recognized, the `font` attribute currently has no effect.

6.3.1 Color-Mapping Functions

Setting the Color Map

The function `ezcscm` is used to create and install a color map with `numcols` entries. (The number of colors should be limited by the value of the EZD variable `numcol`, which defines the number of user-alterable colors.) It is called as follows:

```
ezcscm(numcols)
```

The numbers `1..numcols` then act as color indices into the map. This function should be called before plotting cell arrays to ensure that there will be colors available for the plot.

For the advanced user, the specific color map used can be altered from the default at the time

that the plotting device is opened. See “Setting the Colormap” on page 20 for more information.

Mapping Real Data to Color Indices

The vector-to-color mapping function `ezcmp8` examines a real data vector, determines its maximum and minimum values, `zmin` and `zmax`, and linearly maps the colors `1..ncol` to the data. In other words, the data is partitioned into `ncol` bins and each element that falls in the same bin gets the same color. The resulting color indices are placed into the `clrndx` vector.

The function `ezcmp8ln` performs the same operation, except that the data is mapped to the colors logarithmically; i.e., the logarithm of the data is partitioned into bins with each data element colored according to the bin its logarithm falls into.

The function `ezcmp8nml` maps the data according to a normal distribution. In addition to `zmin` and `zmax`, it computes the mean `zbar` and standard deviation `zsigma` of the data. Values which are over two standard deviations below `zbar` are mapped to color index 1; values over two standard deviations above `zbar` are mapped to color index `ncol`. Intermediate values are mapped in the normal distribution fashion.

In order to accommodate applications for which the data range may change over the course of a computation, these routines also have an input argument `zlim`, which is a two-element real array. If `zlim(1)=zlim(2)`, then `zmin` and `zmax` are computed from the data, as described above. Otherwise, `zmin` is set to the smaller of `zlim(1)` and `zlim(2)`; `zmax`, to the larger value. In this case, data values outside this range are mapped to the appropriate extreme color index.

These functions are called as follows:

```
integer ncol, veclen, clrndx(veclen)
real(Size8) data(veclen), zz(5), zlim(2)
ezcmp8 (ncol, veclen, data, clrndx, zz, zlim)
ezcmp8ln (ncol, veclen, data, clrndx, zz, zlim)
ezcmp8nml (ncol, veclen, data, clrndx, zz, zlim)
```

The arguments have the following meaning:

`ncol` : the number of colors in the color map. Typically, this will be the value of the EZD variable `numcol`. [input: integer]

`veclen` : the length of the data vector. If `data` is dimensioned (`nx`,`ny`), set `veclen`=`nx*ny` and dimension `clrndx` the same as `data`. [input: integer]

`data` : the real data vector. [input: real(Size8) array]

`clrndx` : the resulting color index array. [output: integer array, same shape as `data`.]

`zz` : auxiliary output containing `zz(1)=zmin`, `zz(2)=zmax`, `zz(3)=zbar` (set only by `ezcmp8nml`), `zz(4)=zsigma` (set only by `ezcmp8nml`), `zz(5)` is the mapping type, 0 for linear, 1 for logarithmic, 2 for normal. [output: real (Size8) array]

`zlim` : auxiliary input to allow user-defined limits (see discussion above). [input: real (Size8) array]

Caution: Since `clrndx` and `zz` are output arrays, the names of these variables must be preceded by `&` if these functions are called from Basis, and the type declaration in Basis is `real8`, not `real(Size8)`. See the following example to see how this is done.

Older versions of EZN (before Basis 11.12) provided an `ezclr8`-family of routines, which did not have the `zlim` argument and always used the data limits. In order to provide compatibility for old applications, these are still provided as shells that call their `ezcmp8`-counterparts.

Example

The following example installs a 180-color color map, computes a function on a 100x100 mesh, maps this linearly to color index array `clrz` using the data limits, and displays the result via `ploti`.

```
integer clrNcol=180, nz=100, i, j
ezcscm(clrNcol)
real z(nz,nz), r
do i=1, nz
   do j=1, nz
      r = sqrt(1.0*i*i + 1.0*j*j) + 1e-12
      z(i,j) = sin(r) / r
   enddo
enddo
real8 zlim(2), zz(5)
zlim(1)=0; zlim(2)=0 #To compute zmin,zmax from z.
integer clrz(nz,nz)
ezcmp8(clrNcol, nz*nz, z, &clrz, &zz, zlim)
ploti clrz
```

CHAPTER 7: Mesh-Oriented Commands

Mesh-oriented plots are specific for Lasnex applications. A mesh-oriented command assumes an underlying logically-rectangular two-dimensional mesh. The x-coordinate of the mesh, *xexpr*, and the y-coordinate, *yexpr*, are both two-dimensional real arrays dimensioned (`kmax,lmax`). [`kmax` and `lmax` are internal variables in the EZN package and are Basis variables in Lasnex and in Lasnex dump files.] By convention, *zone* (i,j) is the quadrilateral with upper-right corner (i,j); that is, with diagonally opposite corners (*xexpr*(i-1,j-1), *yexpr*(i-1,j-1)) and (*xexpr*(i,j), *yexpr*(i,j)).

A mesh-oriented command also requires a region map *ireg* as an argument. This is a two-dimensional integer array, also dimensioned (`kmax,lmax`), with *ireg*(i,j) the region number for zone (i,j). The values of *ireg*(1,:) and *ireg*(:,1) are irrelevant. A value of 0 indicates a "void".

The three mesh-defining arrays *xexpr*, *yexpr*, *ireg* have default names `zt`, `rt`, `ireg` respectively. If these variables are specified in the plot command, they must appear before the first *`key=value`* pair. They may be dropped from the right, with missing values replaced by defaults. Thus, "`x,color=red`" is equivalent to "`x,rt,ireg,color=red`".

A mesh-oriented command accepts attribute specifications which specify a subset of the mesh to be plotted by defining values for `krange`, `lrange`, and `region`. The command will plot the subset of the mesh consisting of zones whose indices are in the ranges specified and with region numbers in the region list.

A range specification has the form *(start:stop:inc)*. Unspecified fields in the range are set to default values, e.g. *(:stop:inc), (start::inc), (::inc), (start:), (:stop)*. `krange` specifies a range for the first subscript, and `lrange` specifies a range for the second subscript. The defaults are `krange=(1:kmax:1)` and `lrange=(1:lmax:1)`.

In the specification `region`=*region-list*, *region-list* can be a scalar or vector of integers containing a list of region numbers. The default is `region=all`, meaning all regions.

The attributes `krange`, `lrange` and `region` are "*sticky*", which means that after a mesh-oriented plot specifies a value for an attribute, this attribute value will stay in effect for the following mesh-oriented commands until a new frame or the attribute is reassigned another value.

For example,

```
plotm region=[1,3,5]
       #Mesh plot for regions 1, 3 and 5.
plotc te color=filled
       #The contour plot will be restricted to regions 1,3,5.
nf
```

7.1 plotm: Plotting Meshes, Boundaries, and Regions

Calling Sequence

```
plotm xexpr,yexpr,ireg,<keylist>
plotm <keylist>
plotb <keylist>
```

Description

plotm is a mesh-oriented command. For general information, see the chapter introduction on page 47.

The plotm command plots meshes. If the keyword bnd is set to yes (or 1), only the boundaries of regions are plotted. If specified, *xexpr* is an array of x-axis values, *yexpr* is an array of y-axis values, *ireg* is a region map, and <*keylist*> is a list of optional keywords and values.

If plotm arguments are omitted, they are supplied by using the names in the variables ezcx, ezcy, and ezcireg, respectively. Default values for these names are zt, rt, and ireg.

As a special case, "plotm bnd=1" can be abbreviated plotb.

By convention, the curves connecting nodes are divided into two sets,

k-lines: (*xexpr*(k,:), *yexpr*(k,:)), k=1,...,kmax; and

l-lines: (*xexpr*(:,l), *yexpr*(:,l)), l=1,...,lmax.

The krange and lrange attributes can be given a stride j to cause only every j'th line in that direction to be plotted. The stride is ignored for boundary plots, and ignored in drawing the lines in the opposite direction (that is, the l-lines will have all their pieces even if krange has a stride j, while only every j'th k-line will be plotted).

Optional Attributes

The following optional attributes can be specified with this command. For object attributes, they are local to the command specified; i.e., they are not remembered across commands.

grid, scale, kstyle, lstyle, thick, bnd, color, kcolor, lcolor, mark, marksize, labels, krange, lrange, region, legend

If optional attributes are given on the plot command line, they are specified in the usual form:

key1=value1,key2=value2,...,keyN=valueN

To set an object attribute across commands use the **attr** command. See "Attribute Table" on page 31 for descriptions of the values which can be assigned to these keywords.

The default line style is solid and the default line thickness is 1.0. The default color is the foreground color. To override these defaults, set attributes style, thick, and color respectively.

The attribute `mark` can be used to plot markers at the nodes instead of drawing mesh lines to connect the nodes. This is similar to the command `plot` with the `mark` attribute.

Optional attributes `kstyle` and `lstyle` set the line style for the k-lines and l-lines, respectively. By default, both are set to `solid`. If a style is set to `none`, no lines are plotted in that direction.

Optional attributes `kcolor` and `lcolor` set the line color for the k-lines and l-lines, respectively. If either of these is unset, the color specified by the `color` attribute is used; if both are set, the `color` attribute is irrelevant.

Although it is recognized, the `labels` attribute currently has no effect.

Examples

The following data are used for the examples here and in Sections 7.2 “plotc: Plotting Contours” and 7.3 “plotf: Fillmesh Plot”.

```
#  Define mesh:
integer kmax=25,lmax=35 #Don't make either smaller than 25.
real xr=outer(iota(kmax),ones(lmax)),
     yr=outer(ones(kmax),iota(lmax)),
     zt=5.+xr+.2*ranf(xr),
     rt=100.+yr+.2*ranf(yr)
#  Define region map:
integer ireg(kmax,lmax)=1
     ireg(1,)=0
     ireg(,1)=0
     ireg(2:15,8:12)=2
     ireg(2:15,13:lmax)=3
     ireg(4:7,4:7)=0 #Define an internal void.
integer k2=3,l2=10   #Index of a point in region 2.
#  Define data on the mesh:
real s=1000., z=s*(rt+zt)
     z(4:12,4:10)= z(4:12,4:10)*.9
     z(6,6)=z(6,6)*.9
     z(16:18,18:22)=z(16:18,18:22)*1.2
     z(17,17)=z(17,17)*1.1
```

In the first example, a mesh is plotted with k-lines dashed and l-lines dotted. Here, the displayed mesh has been restricted to lines with k ranging from 1 to 20 and l from 1 to 10. Note that nothing is plotted where the interior void was defined.

```
plotm kstyle=dashed,lstyle=dotted,krange=1:20,lrange=1:10
nf
```

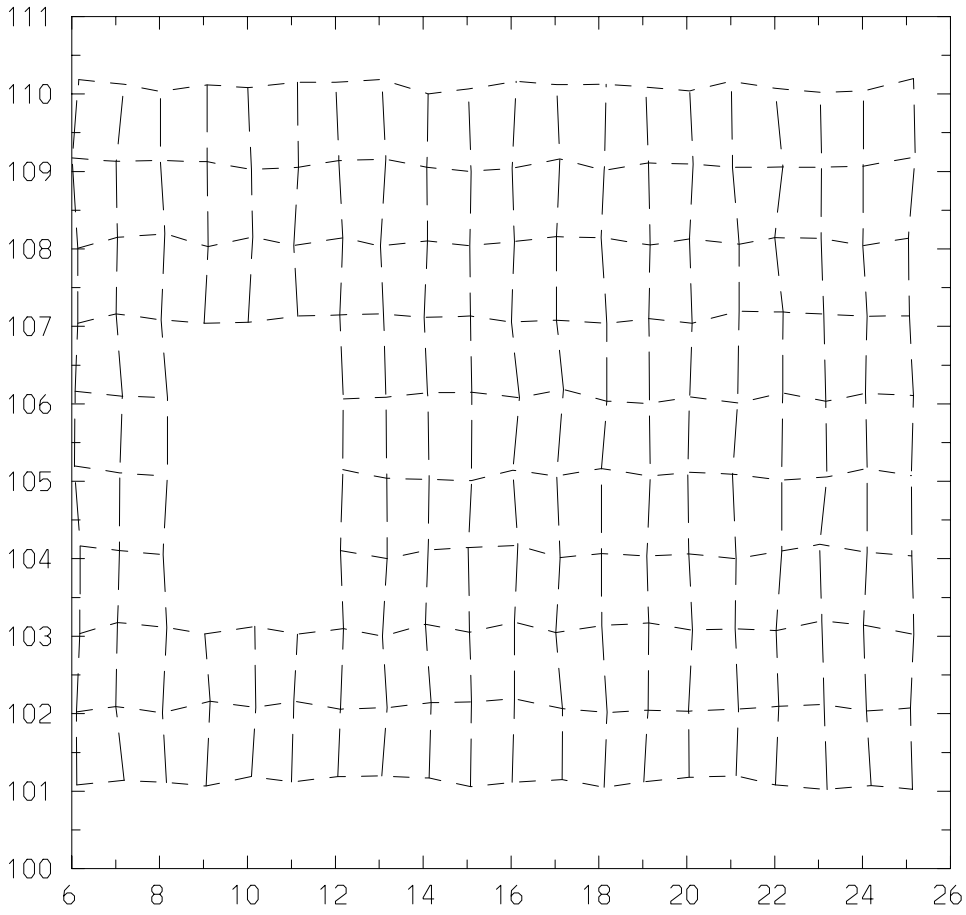
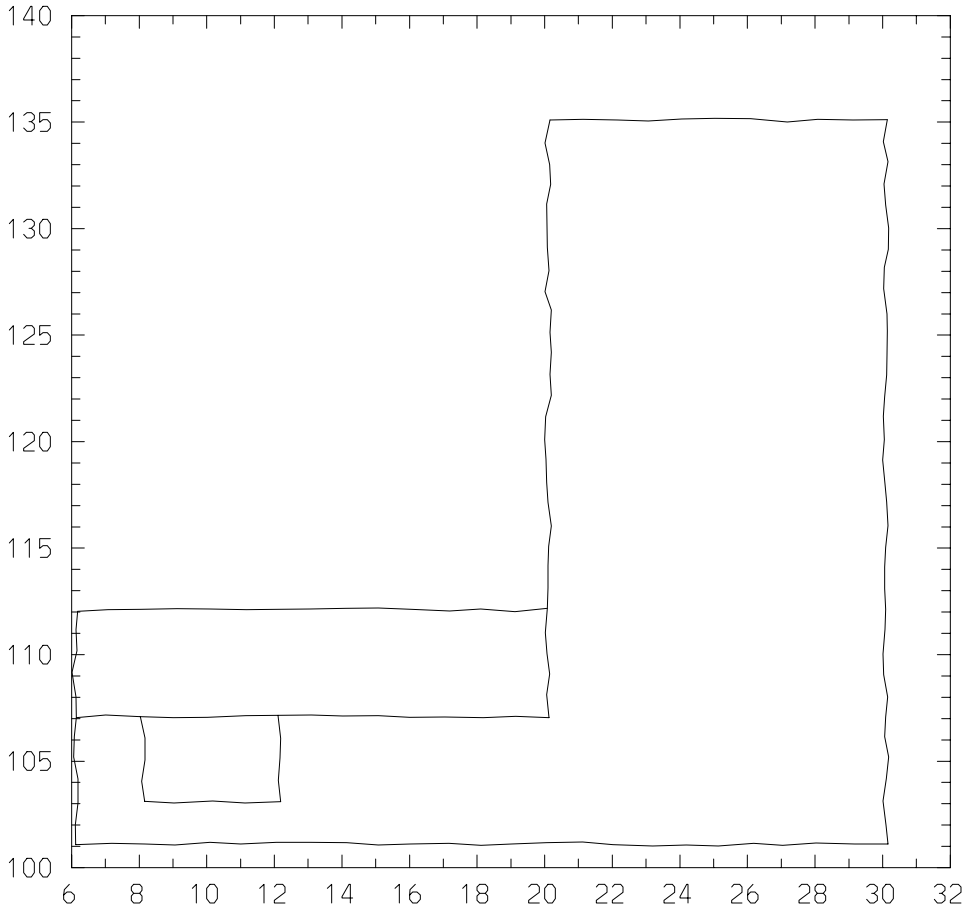


Figure 7.1: Example of Mesh Plot

Here we plot just two regions. Note that the full extent of the mesh is used.

```
plotm bnd=1,region=[1,2] # Plot boundaries of regions 1 and 2.
nf
```

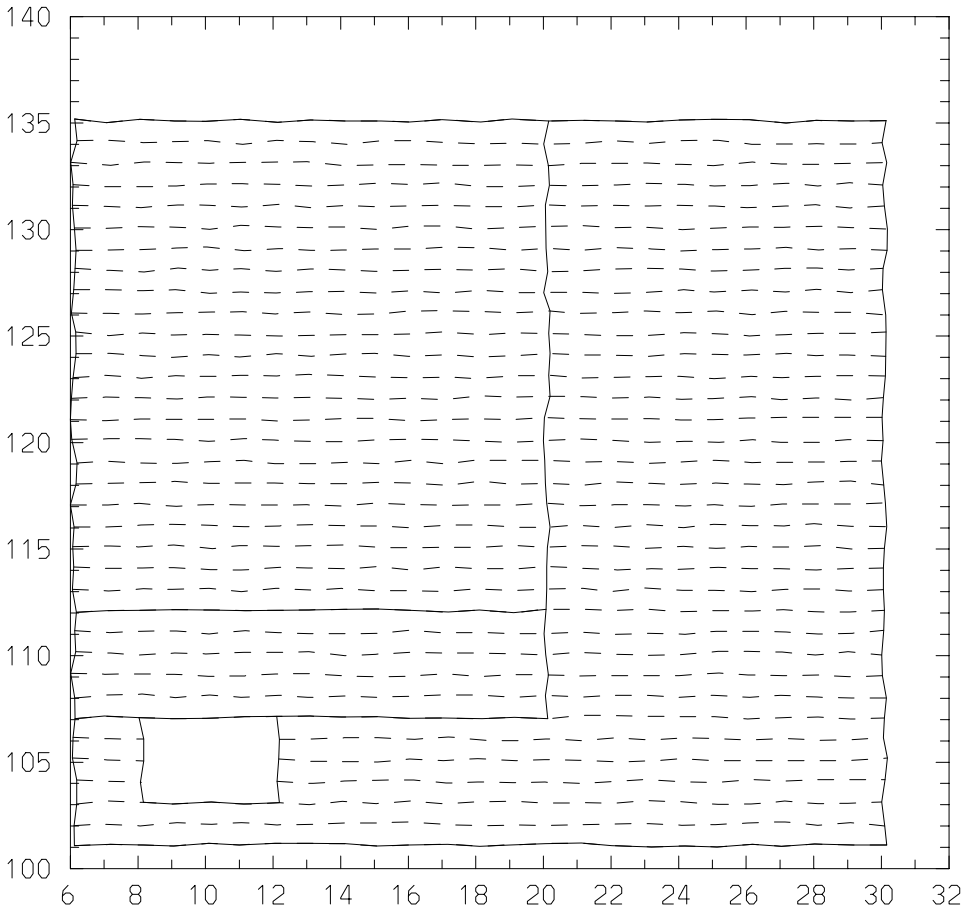



```
plotm bnd=1 region=[1,2]
```

Figure 7.2: Example of Boundaries Plot

And here we plot all region boundaries, and then just the l-lines:

```
plotb                # Plot boundaries.
plotm kstyle=none lstyle=dotted
                     # Plot just the l-lines of the mesh.
nf
```

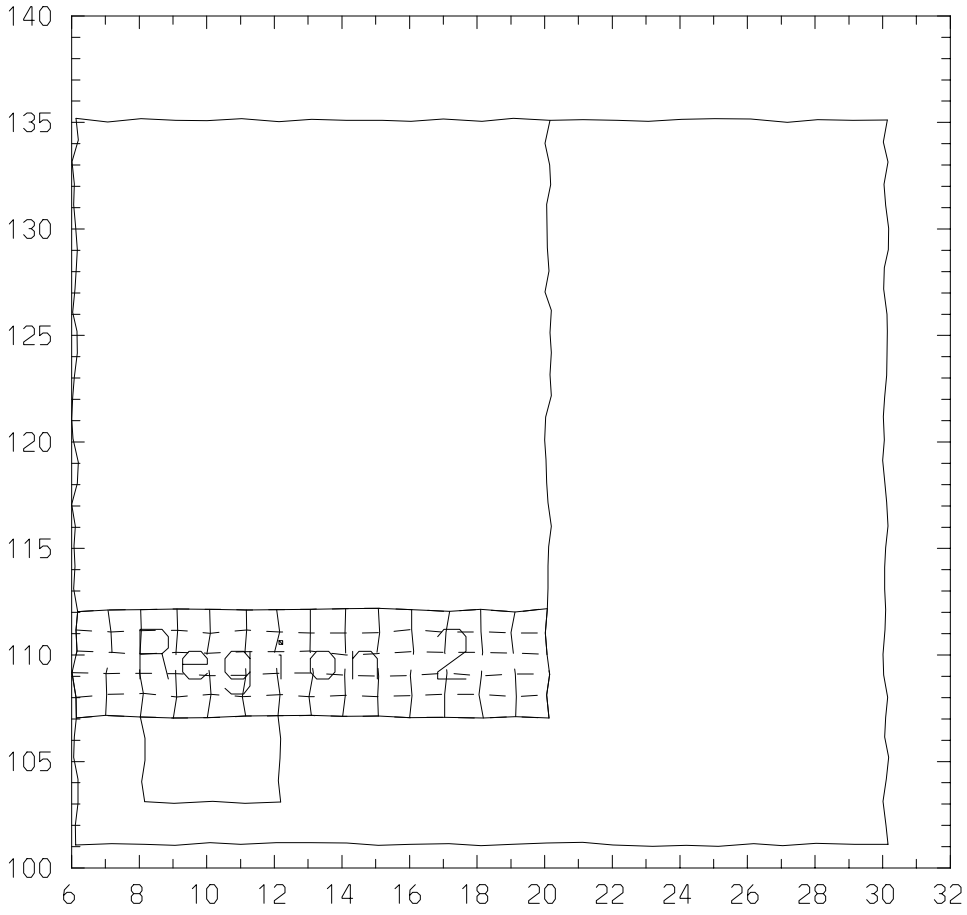



```
plotm bnd=1
plotm kstyle=none lstyle=dotted
```

Figure 7.3: Example of l-lines Plot

Finally, we plot all region boundaries, and mark region 2 with text in it. Note that this looks better on the screen, because the colored mesh lines make the text stand out.

```
plotb                # Plot boundaries
plotm region=2 kstyle=dashed lstyle=dotted color=green
text "Region 2" zt(k2,l2) rt(k2,l2) 32
nf
```




```
plotm bnd=1
plotm region=2 kstyle=dashed lstyle=dotted color=green
text "Region 2" zt(k2,l2) rt(k2,l2) 32
```

Figure 7.4: Example of Plotting a Region with Text

7.2 plotc: Plotting Contours

Calling Sequence

```
plotc fexpr,xexpr,yexpr,ireg,<keylist>
plotc fexpr,<keylist>
```

Description

plotc is a mesh-oriented command. For general information, see the chapter introduction on page 47.

The plotc command plots a contour map of *fexpr* above the mesh described by *xexpr* and *yexpr*. *fexpr* is a two-dimensional array of values dimensioned the same as *xexpr* and *yexpr*. If specified, *xexpr* is an array of x-axis values, *yexpr* is an array of y-axis values, *ireg* is a region map, and <*keylist*> is a list of optional keywords and values. Strides in krange or lrange are ignored by plotc.

If plotc arguments are omitted, they are supplied by using the names in the variables ezcx, ezcy, and ezcireg, respectively. Default values for these names are zt, rt, and ireg.

fexpr can also be the name of a function or macro which, when called with no arguments, returns a two-dimensional array of values of the appropriate shape.

Optional Attributes

The following optional attributes can be specified with this command. For object attributes, they are local to the command specified, i.e. they are not remembered across commands.

grid, scale, thick, style, font, mark, marksize, lev, color, krange, lrange, region, legend, point

If optional attributes are given on the plot command line, they are specified in the usual form:

```
key1=value1,key2=value2,...,keyN=valueN
```

To set an object attribute across commands use the **attr** command. See “Attribute Table” on page 31 for descriptions of the values which can be assigned to these keywords.

Although it is recognized, the font attribute currently has no effect.

Contour Levels, Colors, etc.

The discussion of the command plotz (see Section 6.2 "plotz: Plotting Contours" on page 39) contains a detailed explanation of the way contour levels and colors are specified. The discussion there applies to plotc as well.

The primary difference between plotc and plotz is that the former is a mesh-oriented command. This means that only the plotz mesh data discussion applies to plotc. Furthermore, because of the underlying mesh and the associated region map, the plotc command has the possibility of controlling the subregion over which contours are displayed by use of attributes

`krange`, `lrange`, `region`.

The `plotc` command assumes that the physics quantity *fexpr* is zone-based, or zone-centered, which means that *fexpr*(i,j) is the average value associated with zone (i,j), and the values of *fexpr*(1, :) and *fexpr*(:,1) are irrelevant. Since the contour plot requires data at the mesh points, these values are interpolated to the mesh by simple averaging (as permitted by the subregion specification).

To plot contours of an actual mesh-based, or point-centered, quantity (data at the mesh points), such as `ut` or `vt`, use the specification "`point=yes`" to tell `plotc` that the values are mesh-based. This will avoid the above-mentioned averaging. (The default is `point=no`.) In the case of a mesh-based variable on a mesh containing no interior voids, `plotz` and `plotc` are equivalent. Thus,

```
plotc z,x,y,ireg,point=yes,<keylist>
```

is equivalent to

```
plotz z,x,y,<keylist>
```

where <*keylist*> contains attributes allowed by both commands.

Customizing Contour Plots

Several control variables which may be used to customize the contour plots were discussed in "Contour Control Parameters" on page 40 of the `plotz` description. Some NCAR Conpack parameters can be set to futher customize the contour plots. The routines `cpseti`, `cpsetr`, and `cpsetc` are used to set these parameters. The inquiry routines `cpgeti`, `cpgetr` and `cpgetc` are used to find out the current setting of those parameters. The user should refer to the NCAR document "CONPACK, A Contouring Package" (available on the web at http://ngwww.ucar.edu/ngdoc/ng/supplements/conpack/) for detailed information.

The NCAR Conpack displays dotted lines around "voids". (See Figure 7.5 on page 56 for an example.) These can be eliminated by executing the following calls prior to the `plotc` command:

```
call cpseti ("PAI", -2)
call cpseti ("CLU", 0)
call cpseti ("PAI", -3)
call cpseti ("CLU", 0)
```

The first pair of calls turns off the boundary of an area filled with the "special value" used to indicate missing data. The second pair treats the case of the internal mapping routine returning "out of range", say from the `rt=-1.e8` conventionally used in voids.

Example

The following is an example of using `plotc` with default arguments. The data are as defined before the `plotm` examples, page 49. Note the dotted lines around the internal void.

`plotc z`

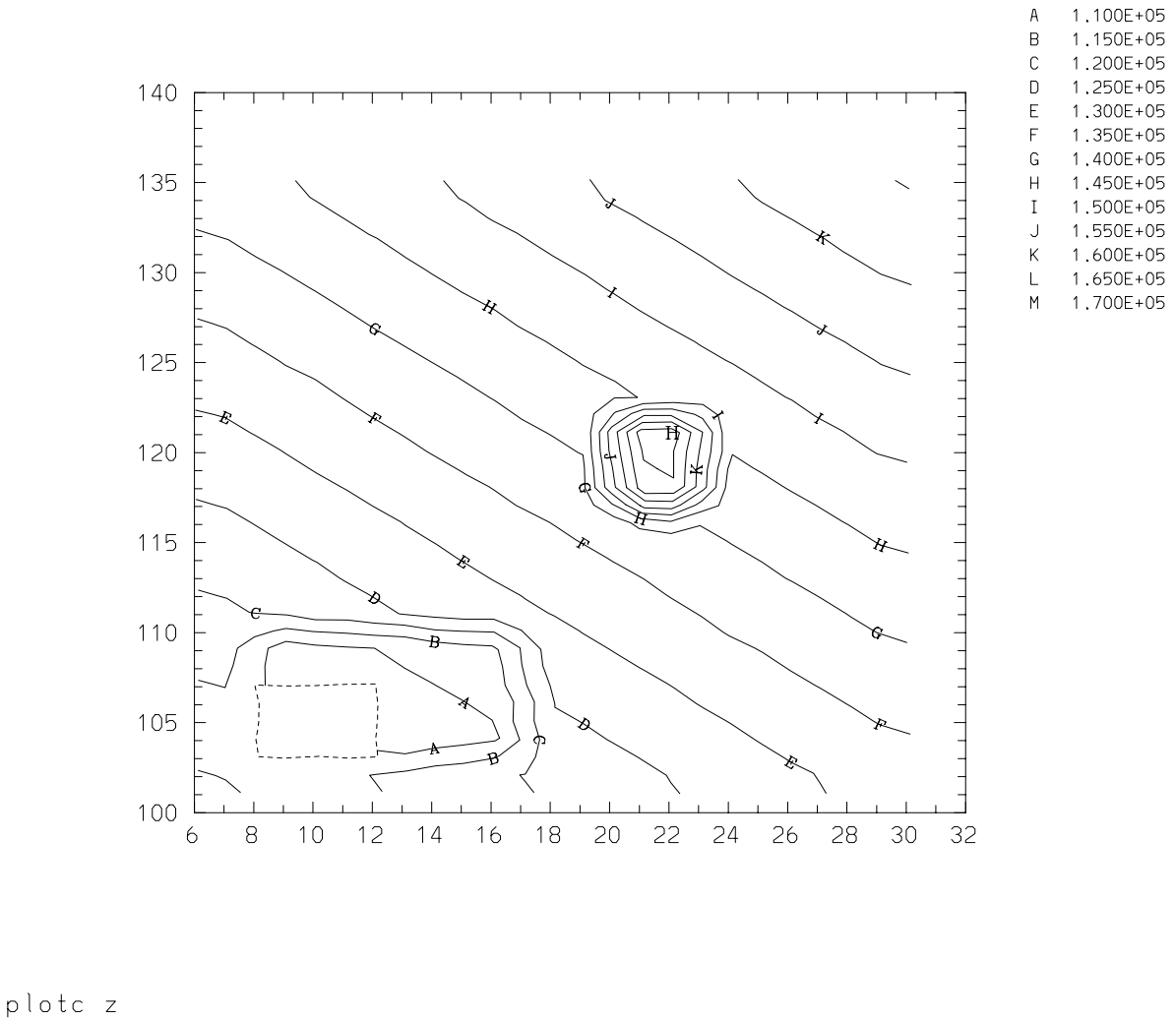


Figure 7.5: Example of Mesh Contour Plot

7.3 plotf: Fillmesh Plot

Calling Sequence

```
plotf pvar,xexpr,yexpr,ireg,<keylist>
plotf pvar,<keylist>
plotf cindex,<keylist>
```

Description

plotf is a mesh-oriented command. For general information, see the chapter introduction on page 47.

The plotf command plots a color-filled mesh which displays the physics quantity *pvar* in the zones of interest with colors. If specified, *xexpr* is an array of x-axis values, *yexpr* is an array of y-axis values, *ireg* is a region map, and <*keylist*> is a list of optional keywords and values.

If plotf arguments are omitted, they are supplied by using the names in the variables ezcx, ezcy, and ezcireg, respectively. Default values for these names are zt, rt, and ireg.

pvar can also be the name of a function or macro which, when called with no arguments, returns a two-dimensional array of values of the appropriate shape.

The colors assigned to the individual zones range from the beginning color in the colormap (after the "named colors" *red, green, blue, yellow*, etc.) to the last color in the colormap. The color varies from low color index to high color index as *pvar* varies from its minimum to maximum values.

The mapping of colors can be *linear*, *logarithmic*, or *normally distributed*. The user can use the attribute cscale to specify the mapping choice. For example, set cscale=log to set the color mapping to logarithmic values of the physics quantity. The default mapping is linear. The normal distribution color mapping (cscale=normal) will map *pvar* values which are over two standard deviations below the mean to the lowest color index, and *pvar* values which are over two standard deviations above the mean to the highest color index. Intermediate *pvar* values are mapped in the normal distribution fashion. A colored annotation on the right side of the frame displays the assignment of colors to the corresponding values of *pvar*.

The attribute zlim=[*zmin*,*zmax*] allows the user to specify limits to be used when mapping physical values to colors (by any of the above-mentioned color scales). If not supplied, the minimum and maximum values in *pvar* are used. The use of zlim allows one to use the same colormap for a series of related plots, such as the time-evolution of *pvar*.

The plotf command also accepts an integer array *cindex* to directly assign color indices to the zones in the mesh. The integer array must be of dimension (kmax,lmax) and contain values between the lowest color index and the highest color index (usually the range 1 to 192). When directly assigned color indices are used, no color annotation will be displayed, because the EZN package has no knowledge of how the color mapping was defined.

Optional Attributes

The following optional attributes can be specified with this command. For *object* attributes, they are local to the command specified, i.e. they are not remembered across commands.

`cscale`, `krange`, `lrange`, `region`, `legend`, `point`, `zlim`

If optional attributes are given on the plot command line, they are specified in the usual form:

`key1=value1,key2=value2,...,keyN=valueN`

To set an object attribute across commands use the **`attr`** command. See “Attribute Table” on page 31 for descriptions of the values which can be assigned to these keywords.

Although it is recognized, the `point` attribute currently has no effect.

Due to the possibility of different color assignment schemes in different regions or with different physics quantities, the `krange`, `lrange`, `region` attributes are made "*non-sticky*"; i.e., the submesh specifications will not be remembered during subsequent fillmesh plots in the same frame. This differs from the effects of `krange`, `lrange`, `region` on the `plotm` command (see Section 7.1 "plotm: Plotting Meshes, Boundaries, and Regions" on page 48).

7.3.1 Fillmesh Level Annotation

When `plotf` is invoked with an array of physics quantities, a display is given to the right of the plot to associate the colors with physical values. Setting `ezcfmkey=off` will cause the frame not to display the fillmesh level annotation. (However the portion of the frame for level annotation is still allocated. To utilize the whole frame without level annotation, the variables `ezccntfr` and `ezcfixed` need be set properly). The default is `ezcfmkey=on`.

The variable `ezcfmfill` specifies either *solid* color fill or *hollow* fill (i.e. just color the border) for each cell containing the numerical annotation. `ezcfmfill=“solid”` (the default) specifies the solid fill; any other value, e.g. `ezcfmfill=“hollow”`, will make a hollow fill.

7.3.2 Color-Mapping Functions

The user who wants to customize the color mapping in a fillmesh plot may wish to see Section 6.3.1 "Color-Mapping Functions" on page 44 for information on the `ezcmp8`-family of functions which produce a `cindx` array directly from the data. Older versions of EZN (before Basis 11.12) provided an `ezcfmc`-family of routines, which did not have the `zz` output array. For compatibility with old applications, these are still provided as shells that call their `ezcmp8`-counterparts.

As an alternative, the user may wish to directly call the routines used by the `plotf` command, namely `ezclrm` for a linear mapping, `ezclrmln` for a logarithmic mapping, or `ezclrmnml` for a normal distribution mapping. These are similar to their `ezcmp8`-counterparts, except that the region map `ireg` is passed as an argument and zones for which `ireg=0` are ignored.

They are called as follows:

```
integer ncol,kmax,lmax,ireg(kmax,lmax),cindx(kmax,lmax)
real(Size8) z(kmax,lmax), zz(5), zlim(2)
```

```
ezclrm (ncol, z, ireg, kmax, lmax, cindx, zz, zlim)
ezclrmln (ncol, z, ireg, kmax, lmax, cindx, zz, zlim)
ezclrmnml (ncol, z, ireg, kmax, lmax, cindx, zz, zlim)
```

The arguments are defined as follows:

`ncol` : the number of colors requested. (Usually use EZD variable `numcol`.) [input: integer]

`z` : an array of physics quantity values. [input: real(Size8) array]

`ireg` : the associated region map (see introductory section, page 47). [input: integer array, same shape as `z`]

`kmax` : the first dimension of the `z`, `ireg`, and `cindx` arrays. [input: integer]

`lmax` : the second dimension of the `z`, `ireg`, and `cindx` arrays. [input: integer]

`cindx` : resulting array of color indices. [output: integer array, same shape as `z`]

`zz` : auxiliary output array containing `zz(1)=zmin`, `zz(2)=zmax`, `zz(3)=zbar` (set only by `ezclrmnml`), `zz(4)=zsigma` (set only by `ezclrmnml`), `zz(5)`is the mapping type, 0 for linear, 1 for logarithmic, 2 for normal. This is used in generating the fillmesh level annotation. [output: real (Size8) array]

`zlim` : the (optional) limitations for the z values; `zlim(1)=zmin`, `zlim(2)=zmax`. (As with `ezcmp8`, if these are equal, use the data limits.) [input: real(Size8) array]

Caution: Since `cindx` and `zz` are output arrays, the names of these variables must be preceded by & if these functions are called from Basis. Note also that the type `real(Size8)` is only recognized in a MPPL Fortran source code. Use `real8` when calling these functions from Basis. (See the final example, below.)

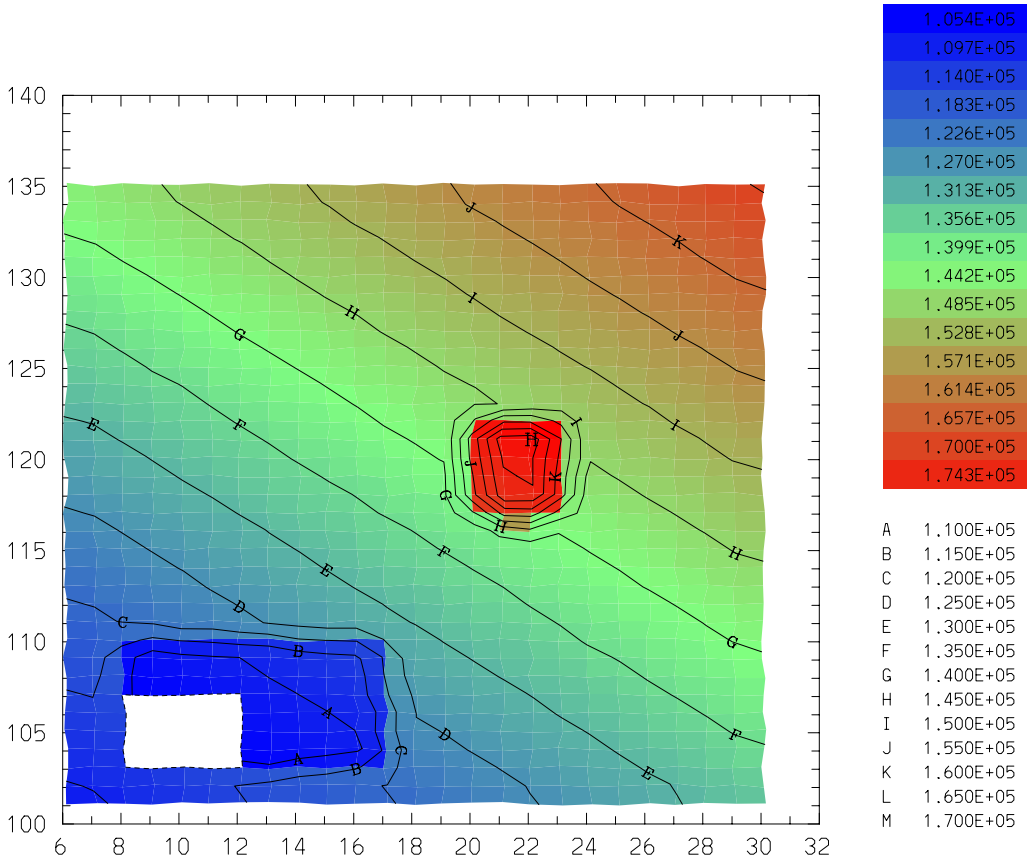
For the advanced user, the specific color map used can be altered from the default at the time that the plotting device is opened. See “Setting the Colormap” on page 20 for more information.

Examples

For our first example, assume the same data as defined before the `plotm` examples, page 49. Note that `plotf` plots nothing in the void, so it has the background color.

```
plotf z
plotc z  # Superimpose contours
nf
```

The result of these commands is shown in Figure 7.6 on page 60. Note the shift in location of the contour annotation from Figure 7.5 on page 56.




```
plotf z
plotc z
```

Figure 7.6: Example of Fillmesh Plot

For our next set of examples, assume that a Lasnex dump file `test1z` has been created, and we want to examine some of its physics variables in Sod. First we do a linearly-scaled fillmesh plot of variable `te`:

```
open test1z
plotf te
nf
```

Next we do a logarithmically-scaled fillmesh plot of variable `ti`:

```
plotf ti cscale=log
nf
```

Finally, we use the color-mapping function `ezclrm` to generate a color index array and plot that. Because we have used the data limits in `zext`, this is essentially the same as the previous plot of `te`, except that it has no color-mapping legend.

```
integer nndx(kmax,lmax)
real8 zz(5),zext(2)
zext(1)=min(te)
zext(2)=max(te)
ezclrm(numcol,te,ireg,kmax,lmax,&nndx,&zz,zext)
plotf nndx
```

7.4 plotv: Plotting Vectors

Calling Sequence

```
plotv xexpr,yexpr,xvexpr,yvexpr,ireg,<keylist>
plotv <keylist>
```

Description

plotv is a mesh-oriented command. For general information, see the chapter introduction on page 47.

The plotv command plots velocity vectors on a mesh. If specified, *xexpr* is an array of x-axis values, *yexpr* is an array of y-axis values, *xvexpr* is the displacement for *xexpr*, *yvexpr* is the displacement for *yexpr*, *ireg* is a region map, and <*keylist*> is a list of optional keywords and values.

If plotv arguments are omitted, they are supplied by using the names in the variables ezcx, ezcy, ezcxv, ezcyv, and ezcireg, respectively. Default values for these names are zt, rt, vt, ut, and ireg. (*Caution*: Note that vt is the velocity in the x-direction; ut, the y-direction.)

A series of arrows from (*xexpr, yexpr*) to (*xexpr+xvexpr*dx, yexpr+yvexpr*dy*) is plotted. The values *dx* and *dy* are chosen so that the maximum extent of an arrow in the corresponding direction is the frame size in that direction multiplied by the vsc attribute. (See also variable ezcvsc.) The default for vsc is .05; this default can be changed by assigning a new value to defvsc.

Optional Attributes

The following optional attributes can be specified with this command. For object attributes, they are local to the command specified, i.e. they are not remembered across commands.

grid, scale, thick, vsc, color, krange, lrange, region, legend

If optional attributes are given on the plot command line, they are specified in the usual form:

```
key1=value1,key2=value2,...,keyN=valueN
```

To set an object attribute across commands use the **attr** command. See “Attribute Table” on page 31 for descriptions of the values which can be assigned to these keywords.

The default line thickness is 1.0 and the default color is the foreground color. To override these defaults, set attributes thick or color, respectively.

Examples

In the first example, the input arrays are explicitly specified. The line thickness of vectors will be 2.0.

```
real vx(10,8),vy(10,8),x(10,8),y(10,8)
```

```
integer i,j, ireg(10,8)
do i=1,10;do j=1,8
   x(i,j)=i; y(i,j)=j
   vx(i,j)=sin(i); vy(i,j)=cos(j)
   enddo;enddo
# Define regions:
ireg(2:5,2:4)=1
ireg(2:5,5:8)=2
ireg(6:10,2:4)=3
ireg(6:10,5:8)=4
plotv x,y,vx,vy,ireg,thick=2.0 # Arguments explicitly specified.
```

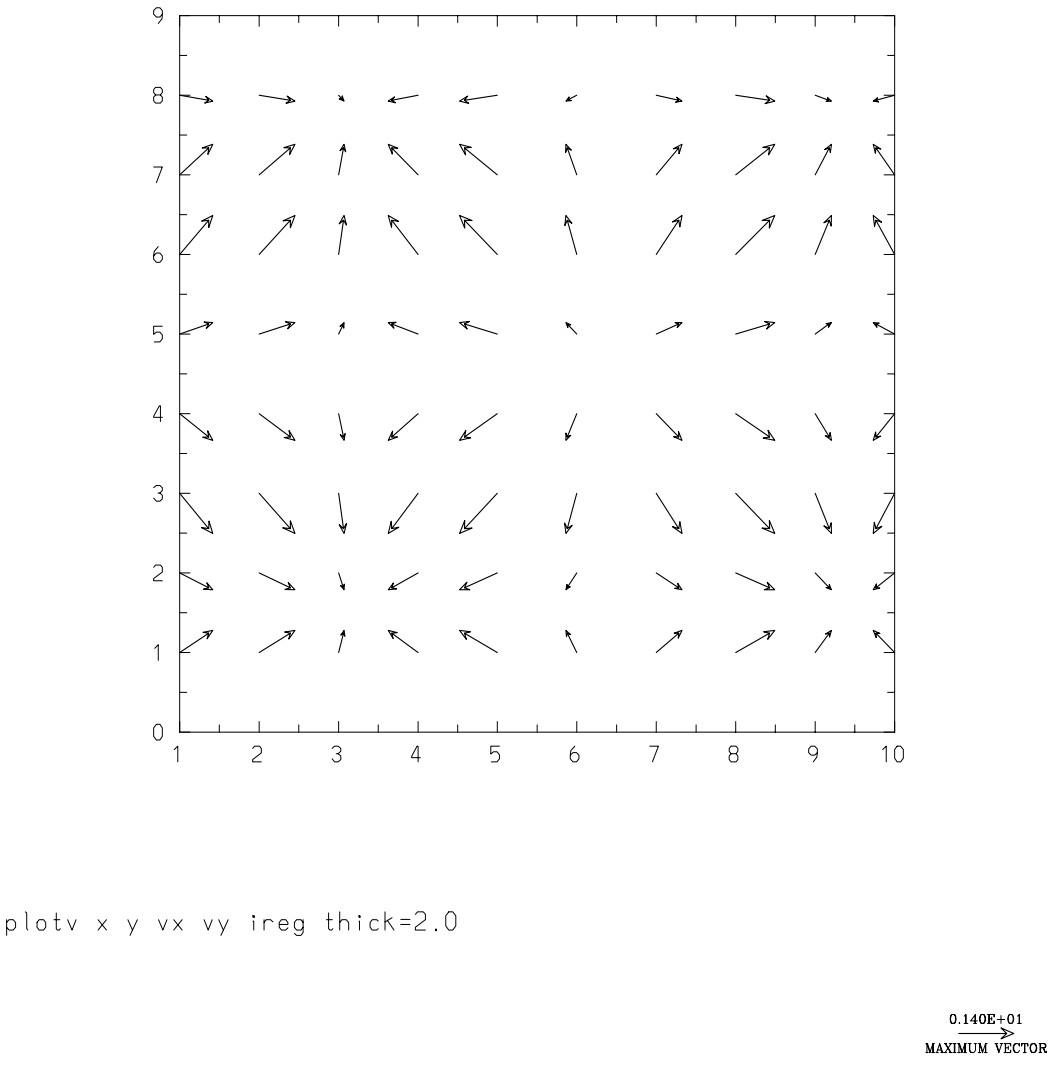


Figure 7.7: Example of plotv

In the second example, the default names `zt`, `rt`, `vt`, `ut`, and `ireg` are used. The displacement vectors are scaled to 0.08 of the frame size. (Note that the vectors are longer and thinner.) Only vectors originating at nodes of zones in regions 1 and 4 are plotted.

```
# Continuation from the last example.
nf
# Set up zt,rt,vt,ut:
real zt=x
real rt=y
real vt=vx
real ut=vy
plotv vsc=.08 region=[1,4]
```

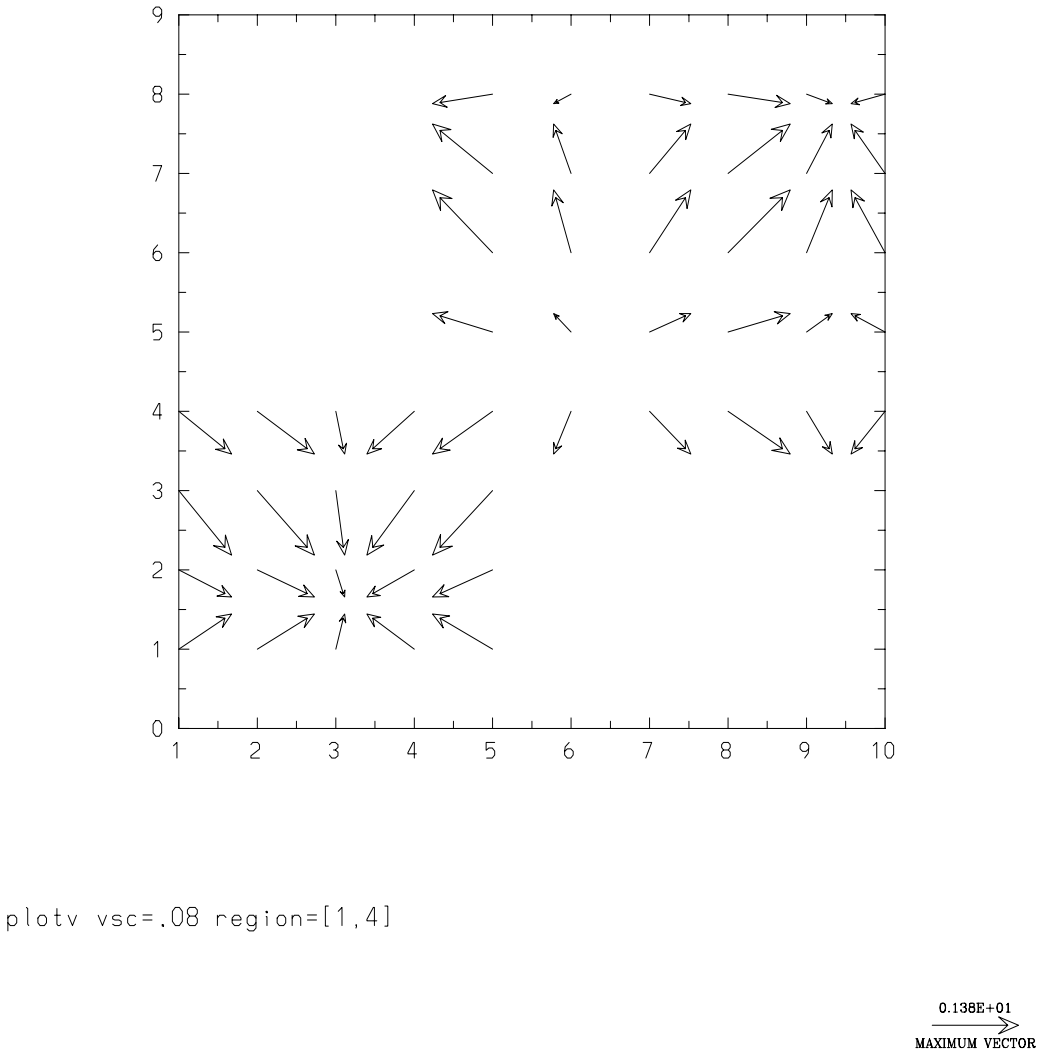


Figure 7.8: Another plotv Example

Customizing Vector Plots

We have already discussed variables `ezcvsc` and `defvsc`, which may be used to control vector plots. Some NCAR Vectors package parameters can be set to futher customize the vector plots. The routines `vvseti`, `vvsetr`, and `vvsetc` are used to set these parameters. The inquiry routines `vvgeti`, `vvgetr` and `vvgetc` are used to find out the current setting of those parameters. The user should refer to the NCAR document "Vectors, A Vector Field Plotting Utility" (available on the web at http://ngwww.ucar.edu/ngdoc/ng/supplements/vectors/) for detailed information.

The NCAR Vectors package displays the magnitude of the largest vector plotted in the lower right-hand corner of the frame, as in the examples. To obtain the value displayed for the maximum vector length, do the following (and don't forget the ampersand):

```
real vecmax
call vvgetr ("VMX", &vecmax)
```

To move the maximum vector display closer to the picture, execute the following calls prior to the `plotv` command:

```
call vvseti ("CPM", -2)
call vvsetc ("MNT", " ")
call vvsetr ("MXX", 0.9)
call vvsetr ("MXY", -.15)
```

The first call is required to change from the default "compatibility mode" so that the values set by the following calls take effect. Without the second call, the magnitude of the minimum vector is displayed near the bottom center of the plot area; this call turns it off. The next two calls set the *x*- and *y*-coordinates of the maximum vector display, relative to the plot area. The default values are approximately 1.02 and -.35, respectively.

```
call vvsetc("MXT", "MY OWN LABEL")
```

will change the "MAXIMUM VECTOR" text to "MY OWN LABEL".

7.5 plotr: Lasnex Rayplots

Calling Sequence

`plotr` *lasernum,<keylist>*

Description

`plotr` is a command which only works in Lasnex or when examining a Lasnex dump file with Sod. The `plotr` command plots rays of the laser number specified. The ray is numbered and if the disposition marks were defined, the special marks will be plotted at the ends of the rays. The ray is plotted with arrows along the path to indicate the direction it travels. *lasernum* is the laser number to be plotted and <*keylist*> is a list of optional keywords and values.

Three control variables `ezcraylab`, `ezcarsp`, `ezcarsz` are provided to customize the appearance of the ray plots. The variable `ezcraylab` can be set to "`on`" or "`off`" to control whether ray labels are plotted or not. The variable `ezcarsp` is a multiplier to the default arrow spacing in the plot. The user specifies this multiplier to extend or to shrink the spacing between arrows. The variable `ezcarsz` is the multiplier to the default arrow size. For example, if `ezcarsz` is set to 0.75, then the arrows plotted will be 75% of the default size.

`plotr` by itself plots the entire ray path, which starts a very long way from the physical mesh. It can be used in conjunction with a `frame` command or with a mesh-oriented command, which will delineate the region of interest.

Optional Attributes

The following optional attributes can be specified with this command. For object attributes, they are local to the command specified, i.e. they are not remembered across commands.

`grid`, `scale`, `thick`, `color`, `legend`

If optional attributes are given on the plot command line, they are specified in the usual form:

`key1=value1,key2=value2,...,keyN=valueN`

To set an object attribute across commands use the **`attr`** command. See "Attribute Table" on page 31 for descriptions of the values which can be assigned to these keywords.

The default line thickness is 1.0. The default color is the foreground color. To override these defaults, set attributes `thick` and `color` respectively.

Ray power

The power of each ray at each point is stored in an array named `rayppow`. This value can be used to color each ray's trajectory depending upon the ray's power at each point. This is done by setting the attribute `color=power`. It is often more useful to color the ray according to the relative power remaining "on" the ray; namely, (raypow(i)-min(raypow))/(max(raypow)-min(raypow)), where the min and max are restricted to the ray being plotted. If `color=relpow`, the color indicates the relative power for each ray. Setting `ezcthickray=on` causes `plotr` to use ray thick-

ness to indicate initial (maximum) power relative to the average maximum power of all rays. However, line thickness is not a very sensitive diagnostic.

When ray power coloring is in effect, a display is given to the right of the plot to associate the colors with physical values. Setting `ezcpwkey=off` will cause the frame not to display the power level annotation. (However the portion of the frame for level annotation is still allocated. To utilize the whole frame without level annotation, the variables `ezccntfr` and `ezcfixed` need be set properly). The default is `ezcfpwkey=on`.

The variable `ezcpwfill` specifies either *solid* color fill or *hollow* fill (i.e. just color the border) for each cell containing the numerical annotation. `ezcpwfill="solid"` (the default) specifies the solid fill; any other value, e.g. `ezcpwfill="hollow"`, will make a hollow fill.

Examples

Assume a Lasnex dump file `test2z` containing laser data has been created, and we want to do post analysis about lasers in Sod:

```
open test2z
plotm
plotr 1
nf

ezcarsp=2. # Set the arrow spacing twice as far as the default.
ezcarsz=0.8# Set the arrow size 80% of the default size.
plotm
plotr color=rainbow# If no lasernum is given, default to 1.
       # color=rainbow makes the rays different colors
       # for easy identification.
```

CHAPTER 8: Polygonal-Mesh Commands

In addition to the logically-rectangular (k,l)-meshes discussed in the previous chapter, EZN also provides some support for arbitrary polygonal meshes (starting with Basis 11.12). A polygonal mesh is simply a collection of polygons defined by two arrays containing the x- and y-coordinates of the polygon vertices. If these are one-dimensional arrays, then a single polygon is defined. If they are two-dimesional arrays, dimensioned (*npts*,*npoly*), then *npoly npts*-sided polygons are being provided. Note that polygons with fewer than *npts* vertices can be included in the collection by repeating the last point as many times as necessary to fill in the *npts* coordinates.

8.1 plotp: Plotting Polygonal Meshes

Calling Sequence

`plotp` *x,y,<keylist>*

Description

The `plotp` command plots a polygonal mesh by filling the polygons with a specified color. If specified, <*keylist*> is a list of optional keywords and values.

Note that no assumptions are made about the connectivity of the collection. If there are overlapping polygons, those appearing later in the list will overplot those plotted earlier.

Optional Attributes

The following optional attributes can be specified with this command. For object attributes, they are local to the command specified; i.e., they are not remembered across commands.

`grid`, **`style`**, **`color`**, **`legend`**

If optional attributes are given on the plot command line, they are specified in the usual form:

`key1=value1,key2=value2,...,keyN=valueN`

To set an object attribute across commands use the **`attr`** command. See “Attribute Table” on page 31 for descriptions of the values which can be assigned to these keywords.

The default color is the foreground color. To override this default, set attribute `color`=*mycolor*. This command has no concept of regions, but you can achieve this same effect by issuing several `plotp` commands on the same frame, using different colors for different regions.

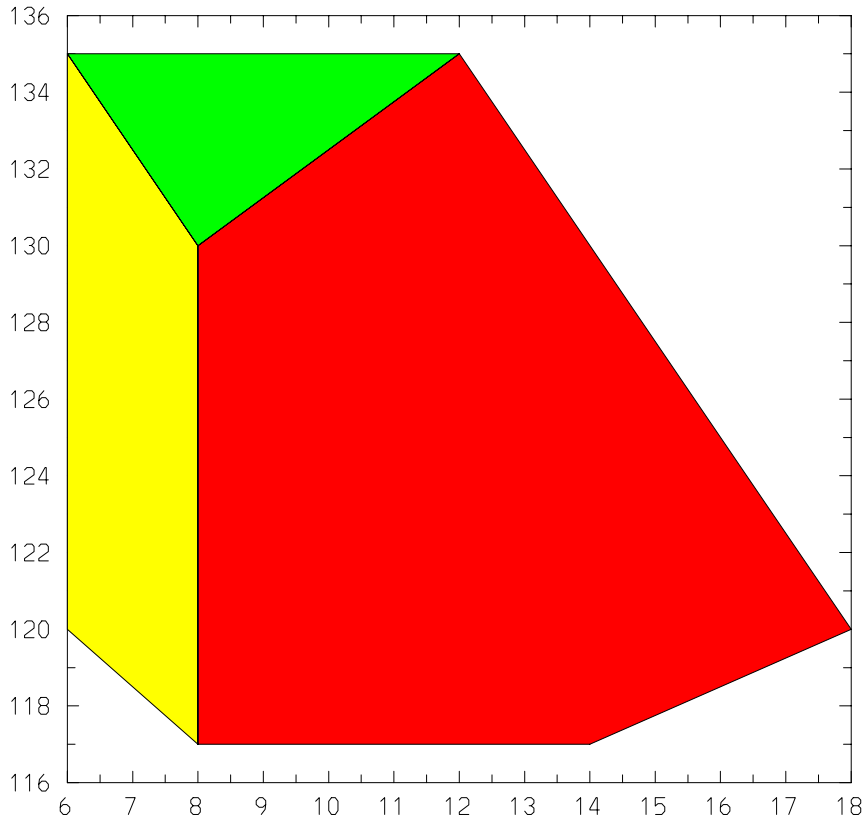
The optional argument `style=`*`mystyle`* can be used to modify the appearance of the plot. The default value for *`mystyle`* is *`solid`*. If any other value, such as *`hollow`*, is given for *`mystyle`*, only the boundaries of the polygons will be plotted. In this case, `plotp` is analogous to the use of `plotm` for a (k,l)-mesh.

Examples

The following code defines a polygonal mesh consisting of a pentagon, a quadrilateral, and a triangle which fill up an irregular hexagon.

```
nf; ezcshow=false
real x5(5,3), y5(5,3)
x5(:,1) = [ 14.,  8.,  8., 12., 18.]
y5(:,1) = [117.,117.,130.,135.,120.]
x5(:,2) = [  8.,  6.,  6.,  8.,  8.]   # Repeat fourth point.
y5(:,2) = [117.,120.,135.,130.,130.]   # Repeat fourth point.
x5(:,3) = [  6.,  8., 12., 12., 12.]   # Repeat third point twice.
y5(:,3) = [135.,130.,135.,135.,135.]   # Repeat third point twice.
# First plot the parts individually, in different colors:
plotp x5(1:5,1) y5(1:5,1) color=red
plotp x5(1:4,2) y5(1:4,2) color=yellow
plotp x5(1:3,3) y5(1:3,3) color=green
sf
# Superimpose full mesh, boundaries only, in the foreground color.
plotp x5 y5 style=hollow
sf
```

The resulting picture appears in Figure 8.1 on page 71.

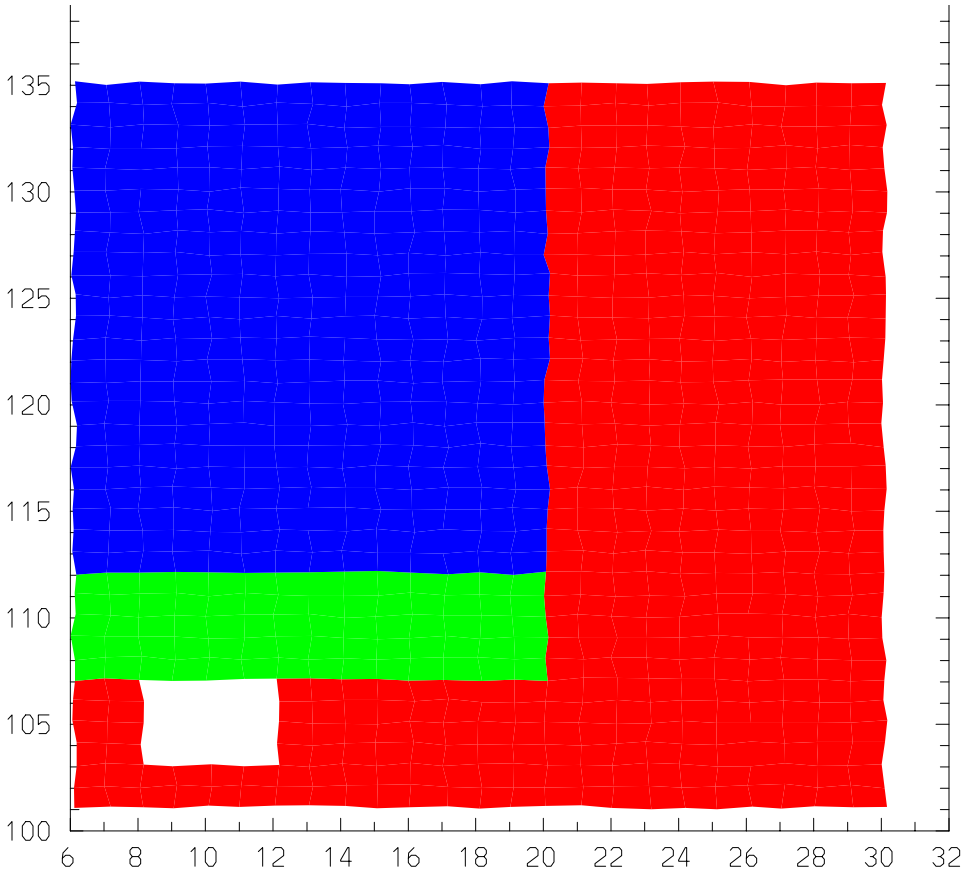



```
plotp x5(1:5,1) y5(1:5,1) color=red
plotp x5(1:4,2) y5(1:4,2) color=yellow
plotp x5(1:3,3) y5(1:3,3) color=green
plotp x5 y5 style=hollow
```

Figure 8.1: Example of Polygonal-Mesh Plot

The second example again assumes the same data as for the `plotm` examples, page 49. We illustrate the use of `plotp` to solid-fill the regions of a mesh with different colors.

```
nf; ezcshow=false
# Read standard Basis utility file to get gather1 function.
read Utilities
# Cycle through the regions, plotting each a different color,
# starting with color 2 in the standard color list.
integer i, kalm, nreg
character*16 mycolor
character*40 mylegend
kalm = kmax*lmax
integer iregx(kalm) = shape( where(ireg==0,-1,ireg), kalm)
# The above set posititons corresponding to ireg=0 to -1.
nreg = max(ireg)
do i=1,nreg
   integer izon = where( iregx=i, iota(kalm) )
   integer nz = length(izon)
   if (nz==0) next  # Omit empty regions.
   mycolor = color(i+1)  # This will work only if < 17 regions.
   mylegend = "Region "//format(i,0)//" is colored "//mycolor
   integer iq(4,nz)
   iq(1,) = izon
   iq(2,) = izon-1
   iq(3,) = izon-kmax-1
   iq(4,) = izon-kmax
   real xq(4,nz) = gather1( shape(zt,kalm), iq)
   real yq(4,nz) = gather1( shape(rt,kalm), iq)
   # Color the i-th region.
   plotp xq yq color=mycolor legend=mylegend
enddo
sf
```




```
Region 1 is colored red
Region 2 is colored green
```

Figure 8.2: Use of plotp to Color Region Map

8.2 plotpf: Polygonal Fillmesh Plot

Calling Sequence

`plotpf` *pvar*`,`*x*`,`*y*`,`*<keylist>*

Description

The `plotpf` command plots a color-filled mesh which displays the physics quantity *pvar* on the polygonal mesh specified by *x,y* with colors. If specified, <*keylist*> is a list of optional keywords and values. This command is analogous to the use of `plotf` for a (k,l)-mesh.

The colors assigned to the individual zones range from the beginning color in the colormap (after the "named colors" *red, green, blue, yellow*, etc.) to the last color in the colormap. The color varies from low color index to high color index as *pvar* varies from its minimum to maximum values.

The mapping of colors can be *linear*, *logarithmic*, or *normally distributed*. The user can use the attribute `cscale` to specify the mapping choice. For example, set `cscale=log` to set the color mapping to logarithmic values of the physics quantity. The default mapping is linear. The normal distribution color mapping (`cscale=normal`) will map *pvar* values which are over two standard deviations below the mean to the lowest color index, and *pvar* values which are over two standard deviations above the mean to the highest color index. Intermediate *pvar* values are mapped in the normal distribution fashion. A colored annotation on the right side of the frame displays the assignment of colors to the corresponding values of *pvar*.

The attribute `zlim=[`*zmin*`,`*zmax*`]` allows the user to specify limits to be used when mapping physical values to colors (by any of the above-mentioned color scales). If not supplied, the minimum and maximum values in *pvar* are used. The use of `zlim` allows one to use the same colormap for a series of related plots, such as the time-evolution of *pvar*.

Optional Attributes

The following optional attributes can be specified with this command. For *object* attributes, they are local to the command specified, i.e. they are not remembered across commands.

`grid`, `cscale`, `legend`, `zlim`

If optional attributes are given on the plot command line, they are specified in the usual form:

`key1=value1,key2=value2,...,keyN=valueN`

To set an object attribute across commands use the **`attr`** command. See "Attribute Table" on page 31 for descriptions of the values which can be assigned to these keywords.

The same annotation is given to the right of the plot as for `plotf`. (See "Fillmesh Level Annotation" on page 58 for control over its appearance.)

CHAPTER 9: Surface Plot Commands

EZN contains a limited number of capabilities for producing wire-frame plots of surfaces defined by data on a rectangular mesh. These are *not* true EZN functions in that they do not add the plots to the EZN display list.

9.1 srfplot: 3-D Surface Plot

Calling Sequence

```
srfplot(x, y, z, nx, ny, view)
```

Description

The `srfplot` call is used to generate a 3-D surface (wire-frame mesh) plot of `z` versus `x` and `y`. In particular, `z` is a matrix of values of size `nx` by `ny`, where `x` and `y` are vectors of length `nx` and `ny`, respectively. The function plotted is then `z(i,j)` = *fcn*(`x(i),y(j)`). The viewpoint of the plot is given by the two-vector `view` where `view(1)` is the angle from the x-axis in the xy-plane and `view(2)` is the angle from the xy-plane. (Angles are in degrees.) Various parameters can be set to control the labels and presentation of the surface plot; see the following subsection.

The `srfplot` subroutine calls the NCAR Graphics routine `SRFACE` and is therefore limited in its interaction with the rest of EZN graphics. In particular, a surface plot cannot share the frame with any other plot, although text may appear. A surface plot cannot be mapped to a quadrant.

Note: The `srfplot` routine is *not* a true Basis Function; it doesn't add the plots to the EZN display list. Commands like "`cgm send`" do not work; you must first activate the desired plotting device(s) before calling `srfplot`.

External Parameters

A number of options to the `srfplot` routine may be controlled through external parameters. These are detailed below.

- Plot Limits – There are 6 external parameters to control plot limits. These are `srfxlo`, `srfxhi`, `srfylo`, `srfyhi`, `srfzlo`, and `srfzhi`. These parameters are used to specify the minimum and maximum of the x-, y-, and z-data. In particular, for a series of surface plots, these may be set to values and then left "frozen" so that plot comparisons can be made. If the value of the `srfautoscal` parameter is set to `false` then the values of the six limit

parameters are used to determine how the plot is scaled. If `srfautoscal` is `true` then the plot is automatically scaled to fill the frame and the limit parameters are ignored. The default is to perform automatic scaling.

- Plot Labels – The parameters `srfxtle`, `srfytle`, and `srfztle` can be set to put titles on the axes. These character strings (maximum length 80) are also used in the legend. If axis labels are not desired, then the parameter `srflabel` can be set to `false`. The legend will still use the axis label parameters regardless of the setting of `srflabel`. The default is to have axis labels.
- Plot Title – The parameter `srftitle` is a character string (maximum length 80) which is used to label the plot, analogous to the super-title for other EZN plots.
- Legend Location – The legend may be located either in the upper right-hand corner of the plot, or the lower right-hand corner of the plot. The default is to put the legend at the top, but this can be overridden by setting `srftopln` to `false`. Note that a relatively long title can intrude into the legend when the legend is located at the top of the plot.
- Skirt – If `srfiskirt` is `true` a "skirt" is plotted around the base of the surface. The height of the skirt can be controlled by parameter `srfhskirt`. The default is no skirt.
- Plot Resolution – For very large data sets, the number of points to be plotted in the x- and y-directions can be specified via the `srfnpx` and `srfnpy` parameters. The default is 100 points in each direction. If `nx>srfnpx` or `ny>srfnpy`, the data are thinned to the resolution specified by these parameters. Requesting too much resolution can produce a very dense plot where details are obscured.

Example

```
integer n=20, i, j
real r, view(2) = [60., 60.]
## Legends and labels for axes:
srfxtle="X Axis"
srfytle="Y Axis"
srfztle="Z Axis"
srftitle="Sombrero Function"
srftopln=false   ## Put legend at the bottom instead of the top.
real x(-n:n) = iota(-n,n)
real y(-n:n) = iota(-n,n)
real z(-n:n, -n:n)
do i = -n, n
   do j = -n, n
      r = sqrt(x(i)**2 + y(j)**2) + 1e-6
      z(i,j) = sin(r) / r
   enddo
enddo
srfplot(x, y, z, 2*n+1, 2*n+1, view)
```

9.2 isoplot: 3-D Isosurface Plot

Calling Sequence

```
isoplot(t, nx, ny, nz, c0, view)
```

Description

The `isoplot` call is used to generate a 3-D surface (wire-frame mesh) approximation to the isosurface *fcn(x,y,z)*=`c0`, where `t` is a three-dimensional array of size `nx` by `ny` by `nz` containing values of *fcn* on a (uniform) rectangular mesh. The viewpoint of the plot is given by the two-vector `view` where `view(1)` is the angle from the x-axis in the xy-plane and `view(2)` is the angle from the xy-plane. (Angles are in degrees.) Various parameters can be set to control the labels and presentation of the isosurface plot; see the following subsection.

The `isoplot` subroutine calls the NCAR Graphics routine ISOSRF and is therefore limited in its interaction with the rest of EZN graphics. In particular, an isosurface plot cannot share the frame with any other plot, although text may appear. An isosurface plot cannot be mapped to a quadrant.

Note: The `isoplot` routine is *not* a true Basis Function; it doesn't add the plots to the EZN display list. Commands like "`cgm send`" to not work; you must first activate the desired plotting device(s) before calling `isoplot`.

External Parameters

A number of options to the `isoplot` routine may be controlled through external parameters. These are detailed below.

- Plot Controls – parameter `isoflg` serves two purposes.

  First, the absolute value of `isoflg` determines which types of lines are drawn to approximate the surface. Three types of lines are considered: lines of constant x, lines of constant y, and lines of constant z. The following table lists the types of lines drawn:

	Plot lines of constant		
`abs(isoflg)`	x	y	z
1	no	no	yes
2	no	yes	no
3	no	yes	yes
4	yes	no	no
5	yes	no	yes
6	yes	yes	no
0, 7 or more	yes	yes	yes

  Second, the sign of `isoflg` determines what is inside and what is outside, hence which lines are visible and what is done at the boundary of the data. For `isoflg`>0, `t` values greater than `c0` are assumed to be inside the solid formed by the drawn surface. For `isoflg`<0, `t` values less than `c0` are assumed to be inside. If the algorithm draws a cube, reverse the sign

of `isoflg`.

The default value is `isoflg`=7 (plot lines of constant x, y, and z).

- Plot Labels – The parameters `isoxtle` and `isoytle` can be set to put titles on the axes. If axis labels are not desired, then the parameter `isolabel` can be set to `false`. The default is to have axis labels.
- Plot Title – The parameter `isotitle` is a character string (maximum length 80) which is used to label the plot, analogous to the super-title for other EZN plots.
- Plot Resolution – For very large data sets, the number of points to be plotted in the x- and y-directions can be specified via the `isonpx`, `isonpy` and `isonpz` parameters. The default is 100 points in each direction. If `nx>iosnpx`, `ny>iosnpy` or `nz>iosnpz`, the data are thinned to the resolution specified by these parameters. Requesting too much resolution can produce a very dense plot where details are obscured.

Example

The following example generates a picture of the 3-D unit ball. The value of `c0` is 0.5 here. Note that the triple loop takes a long time to execute.

```
isotitle="Unit ball in R3"
isoxtle="x"
isoytle="y"
integer nx=10, ny=10, nz=10
real t(-nx:nx,-ny:ny,-nz:nz)
integer i, j, k
real x, y, z
do k = -nz, nz
   z = k / (1.0*nz)
   do j = -ny, ny
      y = j / (1.0*ny)
      do i = -nx, nx
         x = i / (1.0*nx)
         t(i,j,k) = x*x + y*y + z*z
      enddo
   enddo
enddo
isoflg=-7 ## Tell ISOSRF that values < 0.5 are inside.
isoplot(t, 2*nx+1, 2*ny+1, 2*nz+1, 0.5, [15.,15.])
```

CHAPTER 10: Frame Control

There are four commands which control frame actions. The `frame` command sets the limits of the picture frame. The `nf` (New Frame) command is used to begin a new frame. The `sf` (Show Frame) command is used to display the current frame to all active devices. The `undo` command removes a plot command previously issued in a frame.

10.1 frame: Set Frame Limits

Calling Sequence

`frame` *xmin,xmax,ymin,ymax*
`fr` *xmin,xmax,ymin,ymax*

Description

The **`frame`** command sets the limits of the picture frame, which are *frame* type attributes. The `frame` command applies immediately to all plot commands in the frame. `fr` is an abbreviation for "`nf; frame`".

You can supply zero to four arguments. If specified, *xmin* is the minimum value for the x scale, *xmax* is the maximum value for the x scale, *ymin* is the minimum value for the y scale, and *ymax* is the maximum value for the y scale. For each value not specified, the extreme value of the data will be used to calculate the limit. In this case, skipped arguments should be indicated by commas. (Don't put a comma after the last argument you are supplying: Basis' line continuation convention will bite you.) The frame limits will not be retained across frame advances. If a frame already contains objects it will be displayed with these frame limits.

Control Variables and NCAR Autograph Parameters

Some EZN control variables and NCAR Autograph parameters can be used to fine tune the limits of a frame. For example, `ezcextra` controls the extra space below and above the *ymin* and *ymax* when the frame limits are determined by the data extrema. Set `ezcextra=0.` to get rid of this extra space.

The NCAR Autograph package will extend the axes to accommodate labels for the last major ticks in the default case. The parameters "X/NICE.", "Y/NICE." in Autograph can be set to 0 to disable this default behavior. EZN makes `agseti`, `agsetf`, `agsetc`, `aggeti`, `aggetf`, and `aggetc` visible to the user for interactively fine tuning the graphics. For reference, see CHAPTER 11: "Axes, Titles and Text" on page 85, see CHAPTER 15: "Control Variables and Defaults" on page 101, and NCAR Autograph documents for details.

Examples

In the first example, the frame limits are set to the specified values. In the second example, the extreme values for *xmin* and *ymin* are used. Hence, the frame limits are 1,5,1,9.

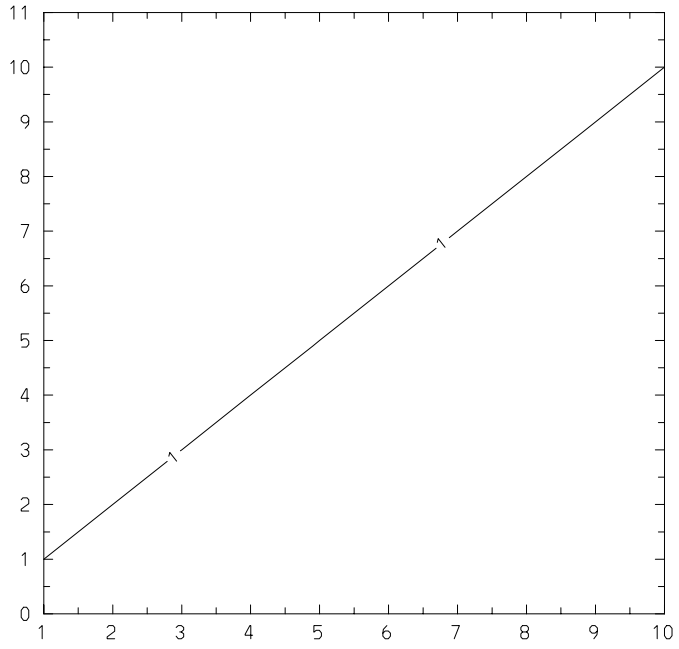
```
ezcshow=true
plot iota(10),iota(10)
frame 2,9,3,7
frame ,5,,9                # xmin,ymin defaulted
frame 2,9                  # ymin,ymax defaulted
```

Since `ezcshow` is `true`, four frames are displayed, as illustrated on the following pages. If `ezcshow` had been set `false`, only three frames would be displayed. The moral is: put the `frame` command first, normally, and use subsequent `frame` commands to plot different views of the same set of objects.

After a picture is displayed, the four values actually used as frame limits are available in the variables `xminu`, `xmaxu`, `yminu`, `ymaxu`. These can be used in calculating arguments to subsequent `frame` commands. In contrast, variables `xmin`, `xmax`, `ymin`, `ymax` will contain the most recent arguments supplied to frame. As an exercise, repeat the above example but type

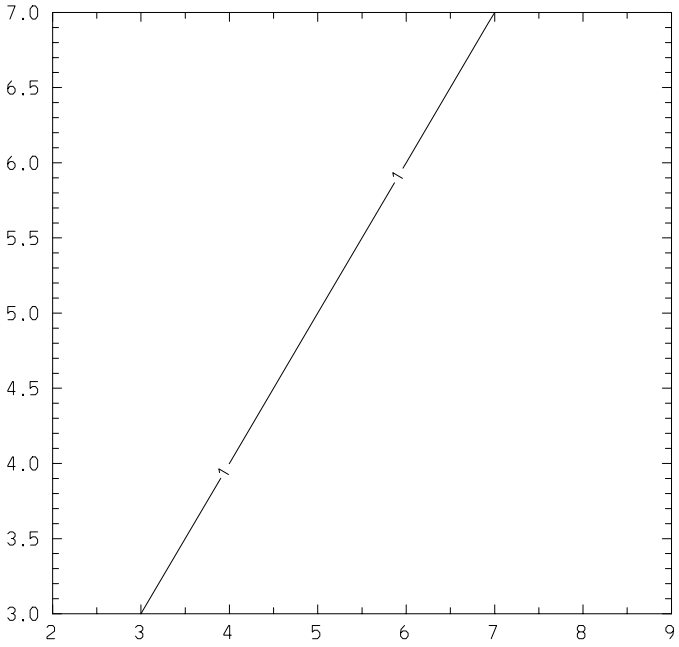
```
xmin,xmax,ymin,ymax
xminu,xmaxu,yminu,ymaxu
```

after each plot command. Note that only the `xmin`,`xmax`,`ymin`,`ymax` values actually given in the previous `frame` command change.



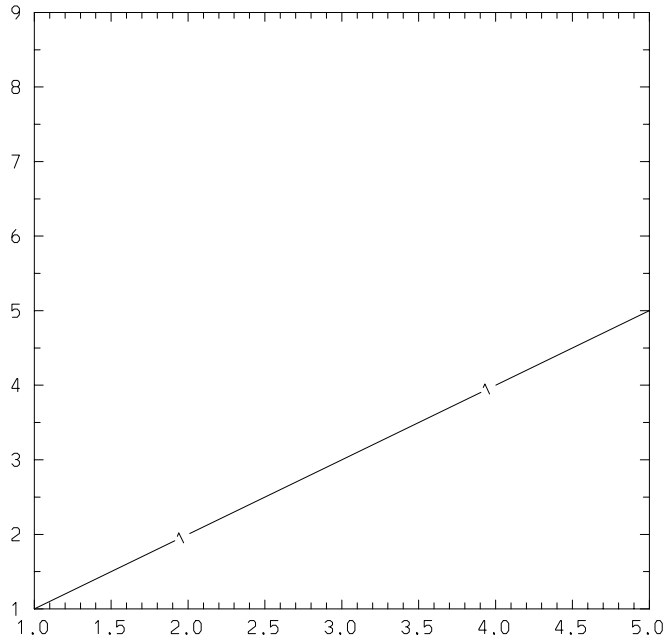

```
1: plot iota(10) iota(10)
```

Figure 10.1: Example of Frame Setting: Default



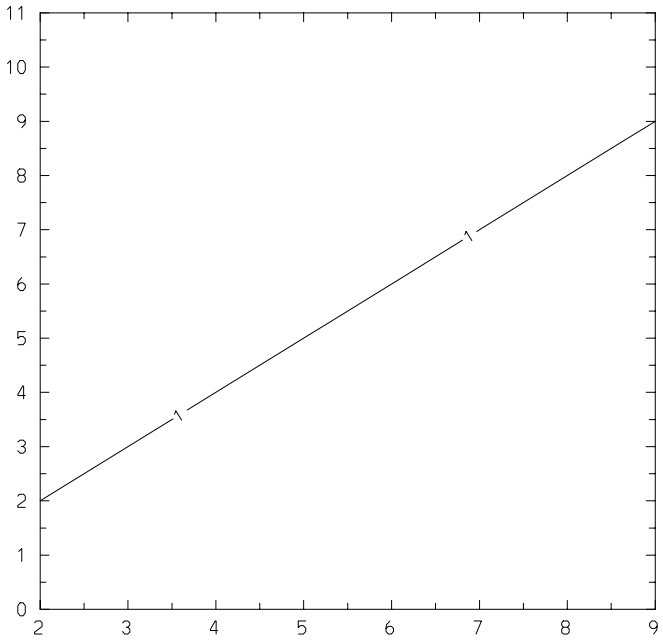

```
1: plot iota(10) iota(10)
```

Figure 10.2: Example of Frame Setting: `frame 2,9,3,7`




```
1: plot iota(10) iota(10)
```

Figure 10.3: Example of Frame Setting: `frame ,5,,9`




```
1: plot iota(10) iota(10)
```

Figure 10.4: Example of Frame Setting: `frame 2,9`

10.2 nf: New Frame

Calling Sequence

```
nf
```

Description

The **nf** command signals that a new frame is to be started. By default, attributes set by the `attr` command are reset to their default values when a new frame is issued.

If variable `ezcreset` is set to false, then the attributes set by the `attr` command remain in effect across frame advances.

What `nf` really does is to close the currently displayed frame. If you are using windows, the effect of `nf` depends on whether or not `ezcshow` is true or false.

- If `ezcshow` is true, you are already looking at the picture, and a `nf` will clear the screen to begin the next one.
- If `ezcshow` is false, you haven't seen the picture yet so `nf` displays it, and this frame will remain displayed, until the next `nf`.

An automatic `nf` is done when the program ends to finish the last frame if required.

Examples

In the default case, the line style is reset across frame advances.

```
ezcreset=true      # (default)
attr style=dashed
plot y,x            # First plot dashed.
plot y2,x2          # Second plot dashed.
nf
plot y3,x3          # Style IS reset to solid (default).
```

In the example below, the line style remains dashed across frame advances.

```
ezcreset=false
attr style=dashed
plot y,x            # First plot dashed.
plot y2,x2          # Second plot dashed.
nf
plot y3,x3          # Style NOT reset across frame advance.
```

(A better way to do this is usually to change the default variables, in this case `defstyle`.)

10.3 sf: Show Frame

Calling Sequence

```
sf
```

Description

The **sf** command displays the current frame to all active devices. The frame is displayed regardless of the value of variable ezcshow. This command is useful when a user wants to control the display of the frame at certain times; i.e., not every time a graphic object is added on a frame (default).

Examples

In the example below, the sf command is used to display the frame after 3 curves have been added. Note that variable ezcshow was set to false. A fourth curve can then be added; had nf been used instead of sf, the first three curves would no longer be in the picture.

```
ezcshow=false
plot y1,x1
plot y2,x2
plot y3,x3
sf          # Force show of current frame.
plot y4,x4
nf
```

10.4 undo: Undo a Plot Command

Calling Sequence

undo *number*

Description

Remove the *number*'th object in the EZN display list. (Each frame EZN has a list of graphic objects when display is requested.) If no argument is given, undo the last graphic object. Some EZN commands do not generate graphic objects in the display list (for example, the frame command), so *cannot be undone* in this way. The easist way is to use the legend as a reference list for undo. For graphic objects whose legend have been suppressed, it is the user's responsibility to figure out which number should be supplied for undo.

CHAPTER 11: Axes, Titles and Text

The axes of a frame are drawn by the NCAR Autograph package. A set of parameters can be set by the user to fine tune the settings of the axes.

There are different ways to plot titles and informational text on a frame. Several variables can be used to control the size, appearance and scope of setting of the titles and text.

11.1 Changing Autograph Parameters

NCAR graphics packages have many parameters which control the appearance of the pictures. One sets these parameters by calling a routine with the name of the parameter and the value. Typically there are three routines for each package used to set real, integer, and character values.

In the NCAR Autograph package, the routines `agsetf("name",fval)`, `agseti("name", ival)`, and `agsetc("name","string")` are used to set the parameters. Note that the "set" call must be made *before* the plot command(s) it is to modify. Use `aggetf`, `aggeti`, or `aggetc` to determine the current setting for a variable. For example, the commands `integer ixnice; call aggeti("X/NICE.",&ixnice)` will return the current value of Autograph parameter `X/NICE.` in Basis variable `ixnice`.

In drawing pictures, EZN makes calls to set the following variables or groups of variables, so they *cannot be set by the user*.

```
WINDOW.
GRAPH.
X.MINIMUM.
X.MAXIMUM.
X.LOGARITHMIC.
Y.MINIMUM.
Y.MAXIMUM.
Y.LOGARITHMIC.
GRID.
BAC.
AXIS/s/CONTROL.
AXIS/s/TICKS/MAJOR/LENGTH/INWARD.
AXIS/s/TICKS/MINOR/LENGTH/INWARD.
LABEL/CONTROL.
DASH/SELECTOR.
DASH/LENGTH.
DASH/PATTERNS/1.
```

In the above, s=LEFT, RIGHT, TOP, or BOTTOM.

The following are the Autograph parameters a user is most likely want to change (note the final period "." is part of the name). A complete list of parameters for controlling axes is given in the NCAR Autograph document (available on the web at http://ngwww.ucar.edu/ngdoc/ng/supplements/autograph/).

TABLE 2. **Autograph Parameters**

Name	default	other settings
X/NICE.	-1	0 disables "nice" x-axis.
Y/NICE.	-1	0 disables "nice" y-axis.
AXIS/s/TICKS/MAJOR/COUNT.	6	*n*>0*:* use *n*+2 to 5*n*/2+4 major tick marks on linear axes.
AXIS/s/TICKS/MINOR/SPACING.	Autograph chooses	*n*<1: no minor tick marks. *n*>=1: *n* minor tick marks per major tick mark.
AXIS/s/NUMERIC/TYPE. (See Autograph documentation for details.)	Autograph chooses the format	0: no numeric labels. 1: scientific notation. 2: exponential notation. 3: "no exponent" notation.
AXIS/s/NUMERIC/EXPONENT.	Autograph chooses	See Autograph documentation. (Used with TYPE.)
AXIS/s/NUMERIC/FRACTION.	Autograph chooses	See Autograph documentation. (Used with TYPE.)
s=LEFT,RIGHT,TOP,BOTTOM		

Example

For example, you may call `agseti("AXIS/BOTTOM/TICKS/MAJOR/COUNT.",3)` to reduce the number of major tick marks on the bottom axis from the default 8-19 to 5-11, and call `agseti("AXIS/BOTTOM/TICKS/MINOR/SPACING.",4)` to introduce four minor tick marks per major tick mark.

11.2 titles: Put Titles on a Plot

Calling Sequence

`titles` *top, bottom, left, right*

Description

Put up to four quoted strings (up to 120 characters each) at the top, bottom, left, and right of the picture, respectively. Each title can also be set individually by assigning a quoted string to the variables `titlet`, `titleb`, `titlel`, and `titler`.

The default value of each title is a blank string. These titles are cleared by `nf`.

The variable `ezctitle` can also be set to a string. This string, referred to as the *supertitle*, will appear on every subsequent frame, usually at the top. The control variable `ezcsuper` can be set `false` to place it at the bottom.

The variables `ezctitfr` controls the size of the titles relative to the graph; the default value is to use 4% of the picture for each title, with 60% of `ezctitfr` used for the supertitle.

For further information, see Section 4.2 "Controlling Layout" on page 23.

11.3 text: Put Text in the Interior of a Plot

Calling Sequence

`text` *string, x, y, nsize, angle, center*

Description

Write the *string* (up to 120 characters) on the plot beginning at coordinates *x*,*y* in User's World Coordinates, using a size argument to the Autograph routine `agpwrt` of *nsize*, at *angle* degrees to the x-axis. The centering of the text with respect to the point (*x*,*y*) is done by passing *center* to `agpwrt`. Usual values for *center* are -1, the default, which centers the left edge at (*x*,*y*); 0, which places the center of the string at (*x*,*y*); and 1, which centers the right edge at (*x*,*y*).

The arguments *nsize*, *angle*, and *center* can be omitted. The defaults are to use text of a minimum size, horizontal, and with the left center edge of the text at the point (*x*,*y*). The text will not be smaller than the size specified by the last `ezcminsz` call (see page 23).

Optional Attributes

The following optional attributes can be specified with this command. For object attributes, they are local to the command specified; i.e., they are not remembered across commands.

`grid`, **`scale`**, **`font`**, **`color`**, **`legend`**

If optional attributes are given on the plot command line, they are specified in the usual form:

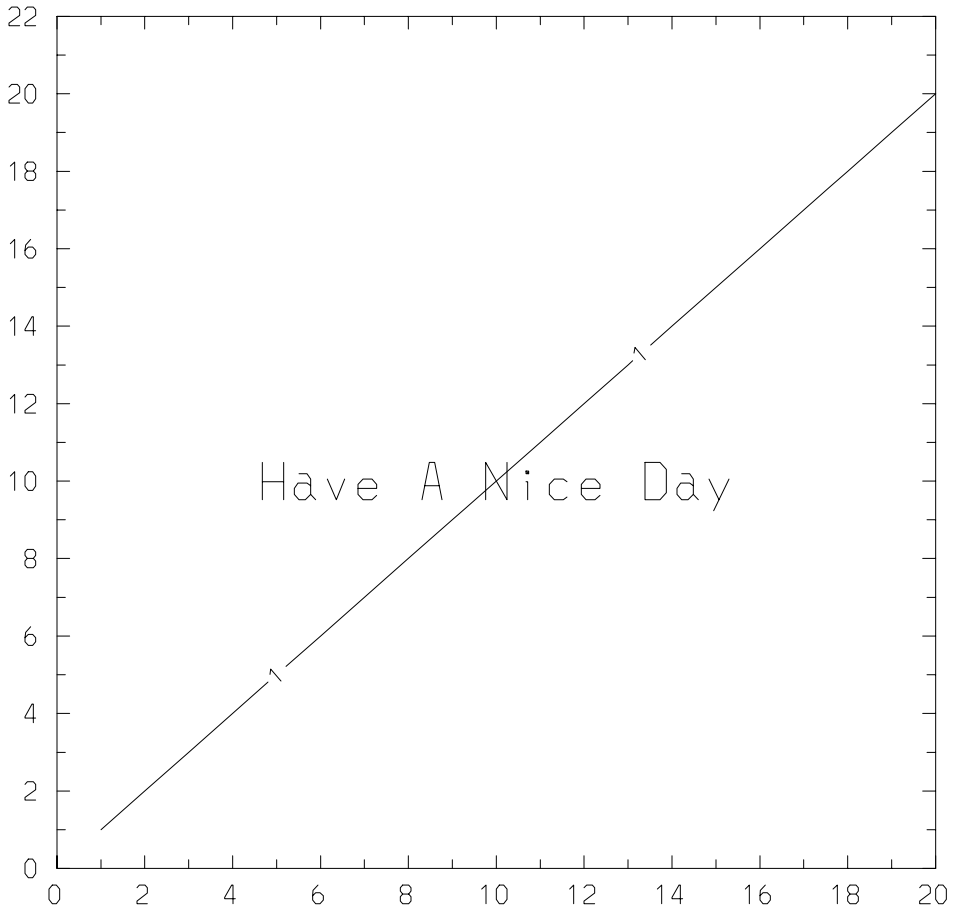
`key1=value1,key2=value2,...,keyN=valueN`

To set an object attribute across commands use the **`attr`** command. See "Attribute Table" on page 31 for descriptions of the values which can be assigned to these keywords.

Although it is recognized, the `font` attribute currently has no effect. (For font control, see Section 11.5 "Text Quality and Optional Fonts" on page 90.)

Example

```
# Example of text command
plot iota(20)
text "Have a Nice Day" 10 10 24 0 0
nf
```




```
1: plot iota(20)
text "Have A Nice Day" 10 10 24 0 0
```

Figure 11.1: Example of Adding Text

11.4 ftext: Put Text Anywhere in a Frame

Calling Sequence

`ftext` *string, x, y, nsize, angle, center*

Description

Write the *string* on the frame beginning at coordinates *x,y* in Normalized Device Coordinates (i.e. the whole frame is a [0.,1.]x[0.,1.] unit square), using a size argument to the Autograph routine `agpwrt` of *nsize*, at *angle* degrees to the x-axis. The centering of the text with respect to the point (*x,y*) is done by passing *center* to `agpwrt`. Usual values for *center* are -1, the default, which centers the left edge at (*x,y*); 0, which places the center of the string at (*x,y*); and 1, which centers the right edge at (*x,y*).

The arguments *nsize*, *angle*, and *center* can be omitted. The defaults are to use text of a minimum size, horizontal, and with the left center edge of the text at the point (*x,y*). The text will not be smaller than the size specified by the last `ezcminsz` call (see page 23).

The differences between the `text` and `ftext` commands are:

- The coordinate system: the `text` command uses *User's World Coordinates* and the `ftext` uses *Normalized Device Coordinates*.
- A string specified with User's World Coordinates will be relocated relative to the frame limits, but a string specified with Normalized Device Coordinates is anchored at the given location on the frame regardless of frame limits changes. Furthermore, a string specified with User's World Coordinates will be clipped to the frame limits.

Optional Attributes

The following optional attributes can be specified with this command. For object attributes, they are local to the command specified; i.e., they are not remembered across commands.

`font`, **`color`**, **`legend`**

If optional attributes are given on the plot command line, they are specified in the usual form:

`key1=value1,key2=value2,...,keyN=valueN`

To set an object attribute across commands use the **`attr`** command. See "Attribute Table" on page 31 for descriptions of the values which can be assigned to these keywords.

Although it is recognized, the `font` attribute currently has no effect. (For font control, see Section 11.5 "Text Quality and Optional Fonts" on page 90.)

11.5 Text Quality and Optional Fonts

On some occasions you may want to use different fonts and/or different quality of the text. EZN provides a routine `ezcstxqu(intq)` for user to change text quality. The input parameter `intq` is an integer with possible values 0 for high quality, 1 for medium quality (default), and 2 for low quality. Higher quality of the text requires more computer resources.

The text quality affects the appearance of labels on the axes, titles, and the text strings specified by the `text` and the `ftext` commands. It currently has no effect on the appearance of the level annotations for contour or fillmesh plots.

Only high quality text (`intq=0`) will allow one to use different fonts; for example, the Greek letters. The optional fonts that came with the NCAR distribution need to be installed on the computer system on which EZN is running; check with your System Manager for the availability of these fonts. In order to specify different fonts by using ASCII characters, the user needs to use function codes embedded in the text string to indicate special letters or symbols. When high quality text is being used, the character size *nsize* is free from the restriction set by `ezcminsz`. Refer to the NCAR Plotchar user's manual for the details.

CHAPTER 12: Stream Output to Graphics

Basis contains a variable `stdplot` which can be used as the unit number in stream output statements. Basis also can be told to redirect most of its output to the graphics package with the "`output graphics`" command. Both of these commands result in calls to a primitive routine `ptext` which writes a line onto the graphics page. The interaction with EZN is as follows:

1. The first such output line will begin on a new frame;
2. Subsequent lines will appear below the previous lines until the frame is full;
3. The number of lines which will fit on a frame is controllable by setting variable `nptext`. The default is 45 lines/frame.
4. The next line to write on can be changed by setting variable `textline`.
5. A `nf` command or an EZN plot command will start a new frame.

Example

As an example, the following puts the Basis version message on the frame together with a message showing the value of a variable and a group named InputStuff:

```
echo=no
output graphics     #Redirect to graphics device.
version             #Print Basis version message.
stdplot <<return    #Print blank line.
stdplot << "This run with alpha = " << alpha
InputStuff
output tty          #Redirect back to tty.
```

CHAPTER 13: Quadrant Mode

You can use the EZN package in *quadrant mode* to place several different pictures on the same frame, and to mix text output with pictures. The routine `ezcsquad` provides general control, while the easy-to-use `ezcquad` routine allows you to put up to four different EZN stream output sessions on the same frame.

`ezcsquad(xmin,xmax,ymin,ymax)` sets the portion of the frame into which the plotting will occur. The four arguments must be in [0.,1.] coordinates. The EZN package is put into quadrant mode, as described below.

`ezcquad(iquad)` is an easier-to-use facility built upon `ezcsquad`. The quadrants are numbered:

```
1  2
3  4
```

The join of 1 and 2 is called 12, the join of 1 and 3 is 13, and likewise for 34 and 24, and finally 1234 is the usual full frame. The command `ezcquad(iquad)` where `iquad` is 1, 2, 3, 4, 12, 13, 24, 34, or 1234, sets the quadrant accordingly by calling `ezcsquad` with appropriate arguments. These appropriate arguments are calculated with respect to a default full frame, but you can call `ezcdquad(xmin,xmax,ymin,ymax)` to set a portion of the screen as the default frame, which is in turn chopped up into quadrants by `ezcquad`.

In quadrant mode, EZN tries to scale everything appropriately. It is usually wise not to use large values for the minimum text size. The number of lines per page for `ptext` output is scaled by the percentage of the vertical space the quadrant occupies. The frame does not advance if too many lines are written; rather, writing returns to the top of the quadrant.

When a subsequent `ezcquad` call is issued, it is as if a `nf` were issued except the frame is not actually advanced; rather, the next plot commands will make a picture in the new quadrant. The delayed-display mode `ezcshow=false` is turned on in quadrant mode.

The `sf` *command should not be used in quadrant mode.*

When the last quadrant is completed, `nf` starts a new frame, and puts you back into non-quadrant mode with a full-screen picture (as defined by the last call to `ezcdquad`). (Note that `ezcshow` is still set to `FALSE`.)

Examples

The following example plots four graphs on one frame:

```
integer i
real x=iota(-5,5)
do i=1, 4
   ezcquad(i)
   plot x**i legend="plot x**"//format(i,0)
enddo
nf
```

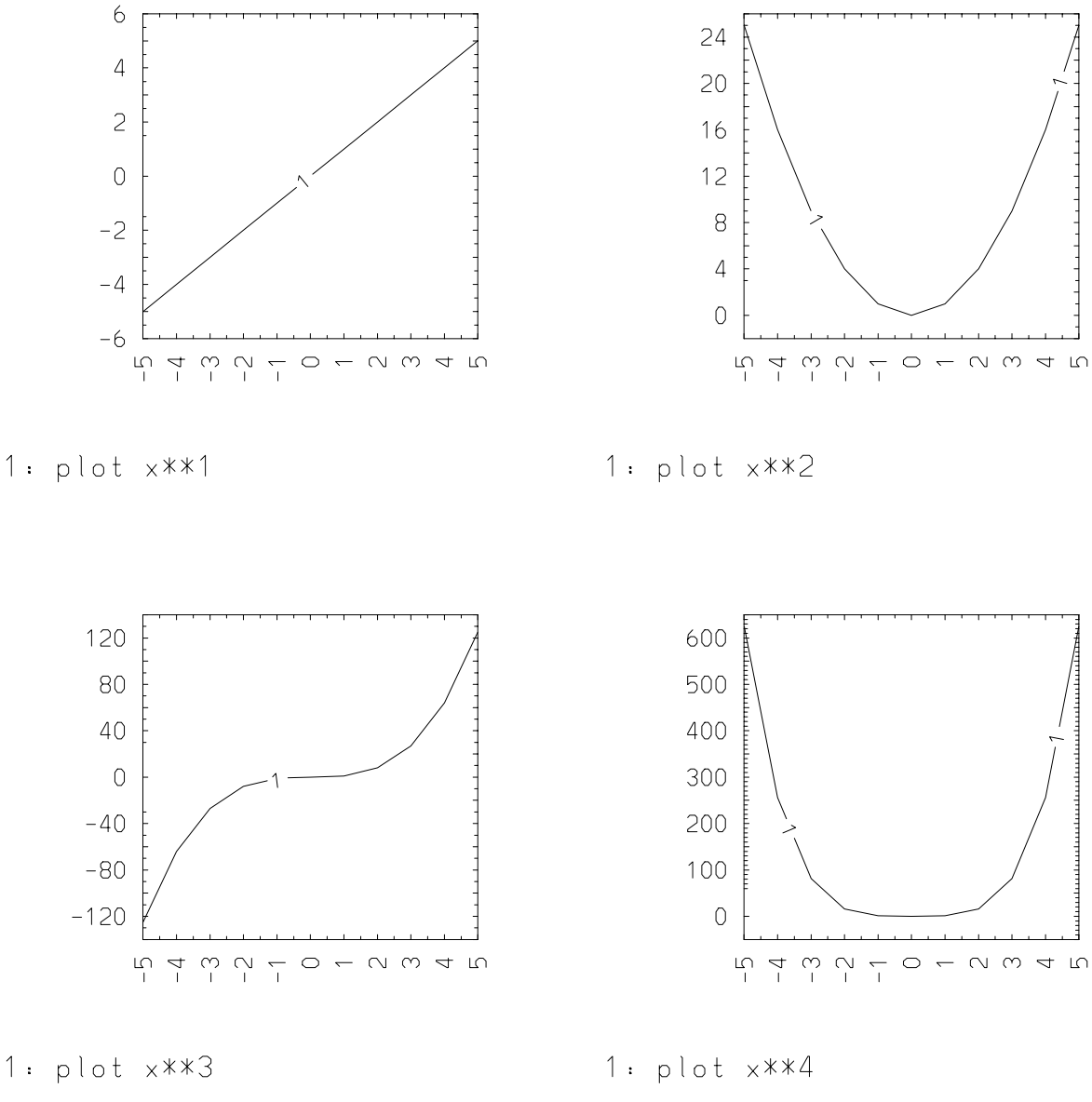
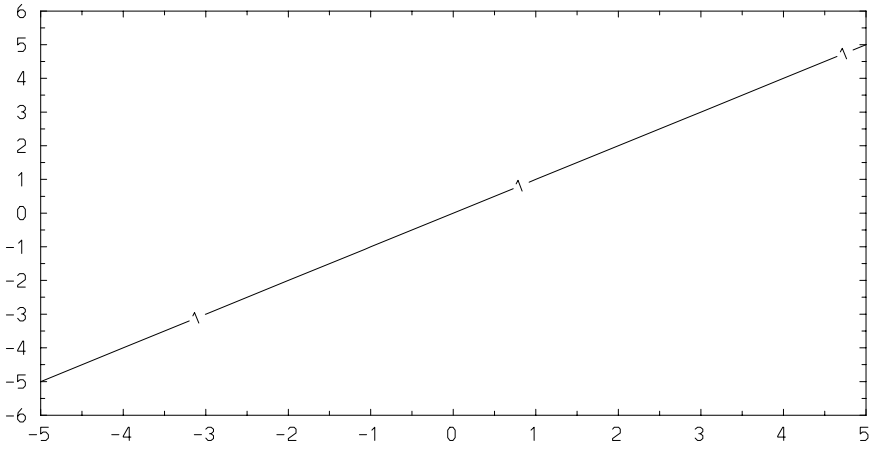


Figure 13.1: Example of Multiple Quadrant Plot

The following example puts a long skinny plot on the top, and some text underneath, with no legend. We turn the echo off so as not to echo the commands themselves in the picture.

```
echo = no
ezcfixed = no
integer i
real x=iota(-5,5)
ezcquad(12)
plot x legend=no
ezcquad(34)
output graphics
nptext = 22 #Want 11 lines to fit exactly.
do i=-5,5
   x(i)
enddo
nf
output tty
```




```
x(i)        =    -5.00000E+00
x(i)        =    -4.00000E+00
x(i)        =    -3.00000E+00
x(i)        =    -2.00000E+00
x(i)        =    -1.00000E+00
x(i)        =     0.00000E+00
x(i)        =     1.00000E+00
x(i)        =     2.00000E+00
x(i)        =     3.00000E+00
x(i)        =     4.00000E+00
x(i)        =     5.00000E+00
```

Figure 13.2: Example of Output Graphics

Note that the quadrant mode does not interact correctly with the send device commands. You will have to use manual control of the output to each device in order to single out a frame for sending to a hardcopy device. Generally, we expect quadrant mode to be used for production output rather than output from interactive exploration.

CHAPTER 14: Interactive Graphics Tools

EZN has several interactive graphics tools that are available for general graphics applications and/or for Lasnex-specific applications. The interactive tools use point and click to interact with the graphics display and to probe the physics quantities. They are applicable only when an X-window is currently open and has an EZN-generated plot displayed on it.

The commands described in this chapter are not part of EZN *per se*. EZN/EZD provides the "hooks" that are necessary to implement them via functions `ezdprobe` and `ezczoom`.

These commands are defined in utility file `interactive.in`, which is generally installed in $BASIS_ROOT/include. To use these commands from Basis or a Basis application, one must first execute command "`read interactive.in`". (This is not necessary for users of Sod or Lasnex, because this file is read at code initialization.)

Once the definitions have been read, the command "`help graphics`" will display a summary of the available interactive graphics commands at the terminal.

14.1 General Graphics Applications

14.1.1 Zoom

The `zoom` function enlarges the picture in a rectangular region bounded by two mouse clicks at the diagonal points. The contents of the graphics display within the selected region will be redrawn to fill the whole frame.

The `zoom` command has the advantage over `frame` with arguments of making it easier to select a specific region of interest. The user will be able to examine the details of the picture by using `zoom` to "zoom in" repeatedly.

14.1.2 Unzoom

The `unzoom` command will return the frame to the previous stage, which gives a chance to select another portion of the frame for further zoom in. The user will be informed when `unzoom` has returned to the original frame. Subsequent `unzoom` commands will have no effect.

A `frame` command without arguments can be used to return the picture to its original size as a short-cut to a series of `unzoom` commands.

14.2 Lasnex-Specific Applications

A set of Lasnex-specific interactive tools can be invoked to mark nodes, to mark a special k-line or l-line, or to highlight a region. One can also request the id of selected zones.

All of these commands assume that a window is open and a mesh-oriented command (see CHAPTER 7: "Mesh-Oriented Commands") has been executed to display a picture in it. They can be used to find information about the mesh associated with the plot.

Caution: These commands require the standard Lasnex mesh variable names to be used. Thus, the mesh must be dimensioned `kmax` by `lmax`, with horizontal axis variable `zt`, vertical axis variable `rt`, and region map `ireg`. The variables `ezdx`, `ezcy`, `ezcireg` are ignored.

A command with a repeated last letter is used to mark a series of points, lines, segments, regions, or zones. Click outside the frame to end the command.

The appearance of markers or highlights can be controlled the same way as the user's plot commands. If the variable `ezcshow` is `true`, then results will be shown immediately; if `ezcshow` has been set to `false`, then they will not be shown until either `sf` or `nf`.

14.2.1 Marking Points

`markp` and `markpp` are used to mark the node(s) pointed to by the mouse. Within the zone where the mouse was clicked, the closest node will be indicated by a circle and its node indices (K,L) will be displayed at the terminal.

The color of the displayed marker(s) can be controlled by preceding the command with "`attr color=`*mycolor*".

14.2.2 Marking Mesh Lines

`markl` and `markll` are the commands to mark one or more k- or l-lines. The user uses the mouse to click two distinct points on a k-line (or an l-line). Then the k-line (or the l-line) containing these two clicked points will be highlighted. As with `markp`, the indices of the node nearest each click will be displayed at the terminal. If the two nodes are not on the same k- or l-line, an error message is given and nothing is plotted.

The color of the highlighted line(s) can be controlled by preceding the command with "`attr color=`*mycolor*" or "`attr kcolor=`*mykcolor* `lcolor=`*mylcolor*". Note: `kcolor` or `lcolor` takes precedence over `color` when all are set and have values other than `fgcolor`.

14.2.3 Marking Mesh Segments

`marks` and `markss` are the commands to mark segments of k- or l-lines. It is used exactly the same as `markl`, but only the portion of the line between the two clicked points will be highlighted.

The color of the highlighted segment(s) can be controlled by preceding the command with an appropriate `attr` command, as for `markl`.

14.2.4 Marking Regions

`markr` and `markrr` are the commands to mark regions bounded by two nodes not on the same k- or l-line. The mesh in the region(s) will be highlighted.

Note: `markr` is actually a function that returns the limits of the region, (*kmin*, *lmin*, *kmax*, *lmax*). To eliminate the extraneous output, use it as "`call markr`".

The color of the highlighted mesh can be controlled by preceding the command with an appropriate `attr` command, as for `markl`.

14.2.5 Marking Zones

`markz` and `markzz` are the commands to mark nodes. When a point is clicked, the zone containing it will be highlighted and its zone indices (K,L) will be displayed at the terminal.

If the variable `verbose=yes`, the (r,z)-coordinates of the clicked point(s) will be displayed along with the zone indices.

The color of the highlighted zone(s) can be controlled by preceding the command with an appropriate `attr` command, as for `markl`.

CHAPTER 15: Control Variables and Defaults

This chapter discusses various variables (also called *parameters*) that are available to control the details of the behavior of the EZN package, as well as routines to query and set parameters.

15.1 EZN Control Variables

There are two groups of variables in the EZN package which control the details of its behavior. The first group, Ezcurve, controls most of the details of the display. The second, EzcurveDefaults, controls the default values of the attributes. The best way to get up-to-date documentation on these is to use the `list` command:

```
output graphdoc
list Ezcurve, EzcurveDefaults
output tty
```

and then print the file `graphdoc`.

15.1.1 Ezcurve Variables

Here are details on many of the variables in group Ezcurve.

`ezcnoplot` If `true`, enter no-plot mode, which makes all graphics commands no-ops. Intended for use in parallel applications in which only the "master" node generates plots. Default: `false`.

`ezcshow` Determines if the current picture is displayed each time it is changed by an EZN command, or only when a *frame* attribute is changed or an `nf` command is issued. By default, `ezcshow=true`.

`ezcreset` Determines if attributes set with the `attr` command are reset to the default values upon a frame advance. If `ezcreset=false`, attributes will remain set across frame advances. By default, `ezcreset=true`.

`ezcextra` If frame limits in the y-direction are not specified, the limits of the data are used, but a small extra space is left on the top and the bottom; this variable contains the factor for the amount of such space (default: .025). Set this variable to 0. to eliminate the space.

`ezcnocx` This logical variable determines whether or not to allow EZN to aver-

age an x array that is one too long to plot a y array against it. Default: `ezcnocx=false`, which means that x can be changed.

`ezcnocy` This logical variable determines whether or not to allow EZN to average a y array that is one too long to plot against the x array. Default: `ezcnocy=false`, which means that y can be changed.

`ezclbshft` Control amount to shift labels for subsequent curve or ray plots to avoid overplotting. An integer in range 0-3. Default: 0.

`ezcfloor` The minimum value for log plots is `ezcfloor` times the maximum data value. The default value is 0., which means to use `r1mach(3)`, approximately 1.E-7.

`dflogstyle` If `true`, log plots have line style `solid` for data above the floor and `dotted` for data which have been promoted to `ezcfloor`. Default: `false`.

`ezctitle` String valued, sets the *supertitle*, put on for every frame; not cleared by `nf`. Default: a blank string.

`ezcsuper` The supertitle is at the top if `true`, at the bottom if `false`. Default: `true`.

`titlet` String valued, sets the *top title* for a frame. Default: a blank string.

`titleb` String valued, sets the *bottom title* for a frame. Default: a blank string.

`titlel` String valued, sets the *left title* for a frame. Default: a blank string.

`titler` String valued, sets the *right title* for a frame. Default: a blank string.

`ezcfixed` If `true`, the plot box will always be the same size regardless of titles and contour labels. If `false`, the plot box expands to its limit. Default: `true`.

`ezctitfr` Fraction of the frame to devote to titles. 0.6 of this is used for `ezctitle`. Default: 0.04.

`ezclegfr` Fraction of the frame to devote to the legend. Default: 0.125.

`ezccntfr` Fraction of the frame to devote to the contour level list. Default: 0.125.

`ezcvsc` Determines the size of the largest vector arrow relative to the frame size for the `plotv` command. See “Attribute Table” on page 31, attribute `vsc`. Default: See `defvsc`, below.

`ezclabel` This string can be set to `"on"`, `"off"`, or `"alpha"` to control the kind of labels to put on contours. The default is single letters (`"alpha"`). A value of `"on"` labels each contour with its value; a value of “`off`” results in no labels at all.

`ezcclf` Format to use to form contour labels when `ezclabel="on"`. Must be a legal Fortran format, including surrounding parentheses, 16 characters maximum, no error checking. Default: `"(1Pe16.2)"`.

`ezcconkey` Control the display of contour level annotation. Set to `"on"` to show the annotation; set to `"off"` for no annotation. Default: `on`.

`ezccksfill` Control the contour level annotation color fill. Use `"solid"` for color fill or `"hollow"` for hollow fill. Default: `hollow`.

`ezcfmkey` Control the display of fillmesh color keys. Set to `"on"` to show the annotation; set to `"off"` for no annotation. Default: `on`.

`ezcfmfill` Control the fillmesh level annotation color fill. Use `"solid"` for color fill or `"hollow"` for hollow fill. Default: `solid`.

`ezcarsz` Multiplier for arrow size in ray plots. Default: 1.0.

`ezcarsp` Multiplier for spacing between arrows in ray plots. Default: 1.0.

`ezcraylab` Control labels in ray plots. Set to `"on"` to plot ray labels, set to `"off"` for no labels. Default: `off`.

`ezcthickray` Control the fat ray option in ray plots. Set to `"on"` to use ray thickness to show relative strength. Default: `on`.

`ezcpwkey` Control the display of ray power color keys. Set to `"on"` to show the annotation; set to `"off"` for no annotation. Default: `on`.

`ezcpwfill` Control the ray power level annotation color fill. Use `"solid"` for color fill or `"hollow"` for hollow fill. Default: `solid`.

`ezciwrk` Amount of integer workspace for Conpack contour routines. Default: 1000.

`ezcrwrk` Amount of real*4 workspace for Conpack contour routines and ARSCAM routines. Default: 5000.

`ezcamap` Amount of integer workspace for ARINAM routines. (Used for filled contour plots and fillmesh plots.) Default: 100000.

15.1.2 Device Control Variables

The EZD group Device_Control also contains variables which the user may find useful for fine control of the detailed behavior of his/her graphics devices.

`ezccgmc` Number of frames to put in a single CGM file. Default: 240.

`ezcpsc` Number of frames to put in a single PS file. Default: 240.

`ezcdisp` Xwindow DISPLAY specification. Overrides the DISPLAY environment variable. Default: a blank string.

`ezcwinwd` Xwindow width, in inches (real). Default: 7.

`ezcwinht` Xwindow height, in inches (real). Default: 7.

`ezcwinsz` Xwindow size specifications (ATC-GKS only): a string value of form
`-dx DX -dy DY -u 0|1|2`
DX, DY are reals; unit specification 0: meters, 1: inches, 2: rasters.

Default: `" -dx 7 -dy 7 -u 1 "`.

`ezcwinlb` Xwindow label specification: a string value to label the window. Default: a blank string

15.1.3 Ezcurve Default Variables

You can change some default settings by assigning new values to the following variables in group EzcurveDefaults. The types and default values are shown for each, in alphabetical order. The notations `_Size4` and `real4` mean a real value in single precision. Note: If reset, these defaults take effect after the next `nf` command, provided `ezcreset=true`. (They will have no effect if `ezcreset=false`.)

`defarrow integer /NO/ -` put arrows on curves? (`YES`=1, `NO`=0)

`defbnd integer /NO/ -` If `YES`, only draw boundaries of regions, not interiors. (`YES`=1, `NO`=0)

`defbot character*LSIZE /" "/ -` title for bottom. (`LSIZE`=120)

`defcolor character*16 /"fgcolor"/ -` normal color

`defcscale character*8 /"lin"/ -` default color index scale. Possible values: `"lin"`, `"log"`, or `"normal"`.

`defgridx character*8 /"off"/ -` grid lines in x direction.

`defgridy character*8 /"off"/ -` grid lines in y direction.

`defksty character*8 /"solid"/ -` style for k-lines.

`deflabel integer /YES/ -` show labels on curves? (`YES`=1, `NO`=0)

`defleft character*LSIZE /" "/ -` title for left. (`LSIZE`=120)

`deflegnd integer /YES/ -` show the legend? (`YES`=1, `NO`=0)

`deflev integer /8/ -` Minimum number of contour levels to choose (NCAR may choose <= twice this); negative means use logarithmic contours.

`deflsty character*8 /"solid"/ -` style for l-lines.

`defmark character*8 /" "/ -` mark -- blank for curves.

`defmarks real4 /1.0_Size4/ -` scale size for marks.

`defpoint integer /NO/ -` are contour z values point-centered?

`defright character*LSIZE /" "/ -` title for right. (`LSIZE`=120)

`defrsq real4 /0._Size4/ -` r-squared parameter for multiquadric algorithm. (Used by random contour plots to interpolate to a grid.) Calculated for you if = 0.

`defscale character*8 /"linlin"/ -` default scale for axes. Possible values: `"linlin"`, `"linlog"`, `"loglin"`, or `"equal"`.

`defstyle character*8 /"solid"/ -` line style.

`defthick real4 /1.0_Size4/ -` thickness of lines.

`deftop character*LSIZE /" "/ -` title for top. (`LSIZE`=120)

`defvsc real4 /0.05_Size4/ -` default value for `vsc`, size of largest vector relative to the frame size.

`ezcx Varname /"zt"/ -` default name for z.

`ezcy Varname /"rt"/ -` default name for r.

`ezcxv Varname /"vt"/ -` default name for v.

`ezcyv Varname /"ut"/ -` default name for u.

`ezcireg Varname /"ireg"/ -` default name for ireg.

15.2 Parameter Access Routines

There are routines for the EZN user to *query* or *set* parameters in EZN and some NCAR packages. These features are for advanced Basis users to futher control their graphics needs.

15.2.1 Query EZN Parameters

There are three routines to query the EZN parameters:

```
ezcgeti("parameter-name",&ival)
ezcgetr("parameter-name",&rval)
ezcgetc("parameter-name",&cval)
```

These routines retrieve the current EZN parameter value even if the parameter is not "`dump`"ed in the variable specification file (i.e., not visible).

The user calls one of these routines based on type of the parameter declared in the " `.v` file", and supplies the parameter name for the query in double quotes. The current value of the parameter will be copied into the integer, the real, or the character string variable specified in the function call.

15.2.2 Set EZN Parameters

There are three routines to set the EZN parameters:

```
ezcseti("parameter-name",ival)
ezcsetr("parameter-name",rval)
ezcsetc("parameter-name",cval)
```

Similar to the query routines, the user may call these routines to set parameters to the values provided in the calling arguments.

15.2.3 Query and Set NCAR Parameters

Similar to the discussion above, user may use `cpgeti`, `cpgetr`, `cpgetc` to query the parameters in the NCAR Conpack package and use `cpseti`, `cpsetr`, `cpsetc` to set its parameters. See “Contour Control Parameters” on page 40 for more information.

`aggeti`, `aggetr`, `aggetc`, `agseti`, `agsetr`, and `agsetc` are the routines for dealing with parameters in the NCAR Autograph package. See “Changing Autograph Parameters” on page 85 for more information.

`vvgeti`, `vvgetr`, `vvgetc`, `vvseti`, `vvsetr`, and `vvsetc` are the routines for dealing with parameters in the NCAR Vectors package. See “Customizing Vector Plots” on page 65 for more information.

Refer to the NCAR manuals (available on the web at URL http://ngwww.ucar.edu/ngdoc/ng/nggenrl/lludoc.html) for complete details.

Subject Index

Symbols

.cgm 15, 17
.cgmlog 15
.ncgm 15, 17
.ps 15
.pslog 15

A

activate device 15
additive model 23
arrow 31
arrows
  on curves 31, 104
  size 103
  spacing 103
asterisk 33
ATC 3, 13
ATC-GKS 1, 15
attr
  command 25, 27, 29
  examples 28–29
attribute list
  ftext 89
  plot 36
  plotc 54
  plotf 58
  ploti 44
  plotm 48
  plotp 69
  plotpf 74
  plotr 66
  plotv 62
  plotz 39
  text 87
attribute table 31
attributes 27
  default 27, 29
  frame 27–28
  non-sticky 58
  object 27–28
  reset 83
  setting 29
  sticky 27, 47
  table of 31
  type 27
Autograph 85, 87, 89, 106
  control parameters 79, 85
axes 23, 85
axis
  control 85
  scale 33, 104

B

background color 20
Basis
  data types 2
  documentation 2
  overview 1
  parser 1
bgcolor 31
bluescale 20
bnd 31, 48
bottom title 87, 102
brownscale 20

C

cell arrays 44
cell-centered 33
CGM 3
cgm
  command 15
  send 15–16
CGM file 15, 17
  frame limit 103
cgm2ncgm 17
circle 33
close 15, 18
  example 16
close all 21
color
  as a device command modifier . . 16
  attribute 31

bgcolor 20, 31
default 39
fgcolor 20, 31
filled 31, 41
fillnl 31
hollow fill 42, 58, 67
normal 104
power 31
rainbow 31
relpow 31
rfill 31
rfillnl 31
solid fill 42, 58, 67
color index scale 104
color indices
mapping real data to 45
colormap 15– 16, 20
bluescale 20
brownscale 20
example 21
greenscale 21
greyscale 20
name 20
pinkscale 21
rainbow 21
setting 44
user-defined 21
color-mapping functions 44, 58
colors
names of 31
commands
attribute setting 25, 27
boundary plots 48
cell array plots 44
contour plots 39, 54
curve plots 35
fillmesh plots 57, 74
frame control 25, 79
general plotting 25, 35
interactive 26, 97
Lasnex rayplots 66
markers 35
mesh plots 48
mesh-oriented 25, 47
polygonal fillmesh 74
polygonal-mesh 25, 69
polygonal-mesh plots 69
quadrant control 26
rayplots 66
region plots 48
surface plotting 25, 75
text plotting 25, 87, 89
vector plotting 62
config 13
Conpack 40– 41, 55, 103, 106
control parameters 41, 55
contour
color filled 41
colors 39, 54
control parameters 40
H and L labels 41
labels 40, 102
format 102
legend 24
level annotation 24, 41, 103
level annotation fill 42, 58, 67, 103
level list. *See* level annotation
levels 28, 32, 40, 54
default 40
mesh 54
plotting 39, 54
style 39– 40
workspace 103
control parameters. *See* variables
control variables. *See* variables
cross 33
cscale 31, 57, 74
default 104
lin 31
log 31
normal 31
ctrans 17

curve
averaging control 101
label control 102
labels 28
plotting 35

D

dashed 33
deactivate device 16
default values 27, 104
changing 29
device
activate 15
deactivate 16
default 16
multiple 18
device command
close 15
colormap 15– 16, 20
list 15– 16
modifier
color 15– 16
mono 15– 16
window-name 16
off 15– 16
on 15
open 15
send 15– 16
slist 15– 16
device type
cgm 15
ps 15
tek 15
tv 15
win 15, 18
display
control 3, 23, 36, 84, 101
default device 16
delayed 23, 93
redirect 15
See also Xwindow
display list 84
dot 33
dotdash 33
dotted 33

E

echo 95
environment variables 1, 3
BASIS_ROOT 1
DISPLAY 1, 15, 103
gksdir 1
MANPATH 1
NCARG_ROOT 1
equal 33
examples
attr 28– 29
attribute resetting 83
axis control 86
cell array 46
colormap 21
contour plot 43
curves 36
frame 80
frame control 84
isoplot 78
labels 37
legend 37
markers 36
mesh plots 50
multiple devices 18
open and send 16
plot 36
plotc 56
plotf 59
ploti 46
plotm 50
plotp 70
plotr 67
plotv 62
plotz 43
quadrant mode 93
ray plot 67

srfplot 76
stream output 91
text 88
Ezcurve 101
EzcurveDefaults 101, 104
EZD 2
EZN 2
ezn.pack 13

F

fat ray option 103
fgcolor 31
filled 31
fillmesh
color keys 103
level annotation 58, 103
plotting 57, 74
workspace 103
fillnl 31
fonts
optional 90
foreground color 20
fr
definition 79
frame 23
attribute type 27– 28
command 25, 79
examples 80
layout 27
limits 79, 101
new 27, 83
set limits 79
show 84
vs zoom 97
with plotr 66
ftext
attributes 89
command 25, 89
compared to text 89
quality of output 90

G

gcaps 17
gist 17
GKS 3
graphics
object 23, 27
redirect output to 91
greenscale 21
greyscale 20
grid 32
no 32
tickonly 32
x 32
xy 32
y 32
gridded data 39

I

ictrans 17
idt 17
index, plotting against 35
interactive
graphics tools 26, 97
mode 23
interactive.in 26, 97
isoplot
command 25, 77
controls 77
example 78
labels 78
resolution 78
title 78
isosurface plots 77

K

kcolor 32, 49
key list 25, 27
keyword 25, 27
keyword. *See also* attribute
k-lines 32, 48– 50

default style 104
krange 32, 47– 48
with plotf 58
kstyle 32, 49

L

labels
attribute 28, 32
example 37
H and L 41
isosurface plot 78
layout of 23
surface plot 76
laser number 66
Lasnex 3
dump file 47, 66
mesh-oriented plots 47
rayplots 66
snapshot 23
layout of frame 23, 27
lcolor 32, 49
left title 87, 102
legend
attribute 28, 32
example 37
layout of 23
surface plot 76
use with undo 84
lev 28, 32, 40
limits
frame 79
surface plot 75
lin 31
line style
default 105
line thickness
default 105
linear
contour levels 40
linlin 33
linlog 33
list 15– 16, 18
l-lines 32– 33, 48– 50
default style 104
log 31
log file 15
logarithmic
contour levels 40
plots 35
floor 102
style 102
loglin 33
loglog 33
lrange 32, 47– 48
with plotf 58
lstyle 32, 49
ltor 33

M

makefile 13
mark 33
asterisk 33
circle 33
cross 33
dot 33
plus 33
x 33
markers
at mesh nodes 39, 49
clipping 35
default 104
default size 104
plotting 35
markl 26, 98
markll 26, 98
markp 26, 98
markpp 26, 98
markr 26, 99
markrr 26, 99
marks 26, 98
marksize 33, 35
markss 26, 98

markz 26, 99
markzz 26, 99
mesh 47
mesh data 39, 54
mesh plots 48
mesh-oriented commands 25, 47
mmm 13
mono 15– 16
multiple windows 18
MultiQuadric 39– 40
mycolormap 21

N

named colors 57
NCAR 3, 13
  ARINAM 103
  ARSCAM 103
  Autograph. *See* Autograph
  Conpack. *See* Conpack
  Plotchar 90
  Vectors. *See* Vectors
NCAR-GKS 15
NCGM file 15, 17
ncgm2cgm 17
network address 15
nf
  command 23, 25, 79, 83
  example 3, 83
  in quadrant mode 93
  with multiple windows 23
  with stream output 91
none 32, 34
non-quadrant mode 93
non-sticky 58
no-plot mode 101
normal 31
normal color 104
Normalized Device Coordinates 89

O

object
  attribute type 27– 28
  graphics 23, 27
off 15– 16
on 15, 18
open 15
  example 16
output
  example 91
  graphics 25, 91
  tty 25, 91

P

parameters. *See* variables
physics quantity
  point-centered 55
  zone-centered 55
pinkscale 21
pkgezn.o 13
plot
  attributes 36
  command 25, 28, 35
  default style 35
  default x 35
  examples 36
  titles 86
  undo 84
plot commands. *See* commands
plotb
  command 25, 48
plotc
  attributes 54
  command 25, 54
  contrasted with plotz 54
  examples 56
Plotchar 90
plotf
  attributes 58
  command 25, 57
  examples 59

ploti
attributes 44
command 25, 44
example 46
plotm
attributes 48
command 25, 48
default style 48
examples 50
markers 49
plotp
attributes 69
command 25, 69
default style 70
examples 70
plotpf
attributes 74
command 25, 74
plotr
attributes 66
command 25, 66
default style 66
example 67
plotv
arrow size 102
attributes 62
command 25, 62
examples 62
plotz
attributes 39
command 25, 39
contrasted with plotc 54
default style 39
example 43
plus 33
pm (plus/minus) 33, 40
point 33, 55
point-centered 55
polygonal-mesh commands 25, 69
polygonal-mesh plots 69
PostScript file 15
frame limit 103
power 31
ps
command 15
send 15, 17
PS. *See* PostScript

Q

quadrant mode 93
defining quadrants 93
examples 93
query parameters 105

R

rainbow 21, 31
range specification 47
ray
arrow size 103
arrow spacing 103
color 31, 66
plotting 66
fat rays 103
labels 103
power 66, 103
power level annotation . . . 67, 103
thickness 31
rayppow 66
real4 104
region 33, 47
list 47
map 47
number 47
plotting 48
with plotf 58
relpow 31, 66
rfill 31
rfillnl 31
right title 87, 102
rsquared 33, 39
default 104
rtol 33

S

scale 33
- equal 33
- linlin 33
- linlog 33
- loglin 33
- loglog 33

scattered data 39– 40
send 15– 16
- example 16
- with quadrant mode 96

set parameters 105
sf
- command 25, 79, 84
- example 3, 84
- with ezcshow 23
- with quadrant mode 93

Size4 104
skirt 76
slist 15– 16, 18
snapshot (Lasnex) 23
Sod 3
solid 33
srfplot
- command 25, 75
- example 76
- labels 76
- legend 76
- limits 75
- resolution 76
- skirt 76
- title 76

steerable applications 1
sticky 27, 47
stream output 93
stride 48, 54
style
- attribute 33
- contour 39
- curve 35
- dashed 33
- dotdash 33
- dotted 33
- ltor 33
- none 34
- pm 33
- rtol 33
- solid 33

supertitle 23, 87, 102
surface plots 75
surface plotting 25

T

tek
- command 15

Tektronix 15, 17
text
- attributes 87
- command 25, 87
- compared to ftext 89
- example 88
- high quality 90
- plotting 85
- quality of output 90
- size 23

thick 34– 35
tickonly 32
title
- bottom 87, 102
- isosurface plot 78
- layout of 23
- left 87, 102
- plotting 85
- right 87, 102
- supertitle 23, 87, 102
- surface plot 76
- top 87, 102

titles
- command 25, 86
- plotting 85– 86

top title 87, 102

tv 15

U

undo
command 25, 79, 84
unzoom 26, 97
User's World Coordinates 89

V

variables
Autograph control 79
Conpack control 41
contour control 40, 55
default values 101, 104
EZN control 101
query values 105
set values 105
user-settable 101
Vector control 65
visible 105
vector plots 62
Vectors 65, 106
control parameters 65
visible variable 105
void 47, 50, 56
voids
boundary of 55
vsc 34, 62, 102
default 105

W

win
close 18
close all 21
command 15, 18
list 18
on 18
slist 18
window
active 18
clear 19
height 18, 103
multiple 18, 23
name 15– 16, 18
size 18, 103
width 18, 103
wire-frame plots 75

X

x-averaging control 35
Xwindow 3, 15
title bar 16

Y

y-averaging control 35

Z

zlim 34, 57, 74
zone 47
zone-centered 33, 55
zoom 26, 97

Index of Variables and Subprograms

A

aggetc 79, 85, 106
aggetf 79, 85
aggeti 79, 85, 106
aggetr 106
agpwrt 87, 89
agsetc 79, 85, 106
agsetf 79, 85
agseti 79, 85, 106
  example 86
agsetr 106
ARINAM 103
ARSCAM 103

B

BASIS_ROOT 1

C

cpgetc 41, 55, 106
cpgeti 41, 55, 106
cpgetr 41, 55, 106
cpsetc 40, 55, 106
cpseti 40, 55, 106
cpsetr 40, 55, 106

D

defarrow 104
defbnd 104
defbot 104
defcolor 104
defcscale 104
defgridx 104
defgridy 104
defksty 104
deflabel 104
defleft 104
deflegnd 104
deflev 40, 104
deflsty 104
defmark 104
defmarks 104
defpoint 104
defright 104
defrsq 104
defscale 104
defstyle 83, 105
defthick 105
deftop 105
defvsc 62, 105
dflogstyle 102
DISPLAY 1, 15, 103

E

ezcamap 103
ezcarsp 103
ezcarsz 103
ezcblue 21
ezccgmc 103
ezccksfill 42, 103
ezcclf 102
ezccntfr 23, 42, 102
ezcconkey 42, 103
ezcdisp 15, 103
ezcdquad 93
ezcextra 79, 101
ezcfixed 23, 42, 102
ezcfloor 102
ezcfmc-family (archaic) 58
ezcfmfill 58, 103
ezcfmkey 58, 103
ezcgetc 105
ezcgeti 105
ezcgetr 105
ezcgreen 21
ezcireg 48, 54, 57, 62, 105
ezciwrk 103
ezclabel 40, 42, 102
ezclbshft 102
ezclegfr 23, 102
ezclr8-family (archaic) 46
ezclrm 58– 59
ezclrmln 58– 59

ezclrmnml 58– 59
ezcminsz 23, 87, 89– 90
ezcmp8 45
ezcmp8ln 45
ezcmp8nml 45
ezcnocx 35
ezcnocy 35
ezcnoplot 101
ezcnox 101
ezcnoy 102
ezcpsc 103
ezcpwfill 67, 103
ezcpwkey 67, 103
ezcquad 26, 93
ezcraylab 103
ezcred 21
ezcreset 27, 29, 83, 101
  example 83
ezcrwrk 103
ezcscm 44
ezcsetbb 20
ezcsetbw 20
ezcsetc 105
ezcseti 105
ezcsetr 105
ezcshow 3, 23, 27, 83, 101
  example 36, 84
  in quadrant mode 93
ezcsquad 26, 93
ezcstxqu 90
ezcsuper 23, 87, 102
ezcthickray 103
ezctitfr 23, 87, 102
ezctitle 23, 25, 87, 102
ezcvsc 62, 102
ezcwinht 18, 103
ezcwinlb 104
ezcwinsz 103
ezcwinwd 18, 103
ezcx 48, 54, 57, 62, 105
ezcxv 62, 105
ezcy 48, 54, 57, 62, 105
ezcyv 62, 105
ezczoom 97
ezdprobe 97

G

gksdir 1

I

ireg 47– 48, 54, 57, 62, 105
isoflg 77
isolabel 78
isonpx 78
isonpy 78
isonpz 78
ISOSRF 77
isotitle 78
isoxtle 78
isoytle 78

K

kmax 47, 57

L

lmax 47, 57

M

MANPATH 1

N

NCARG_ROOT 1
nptext 91
numcol 44– 45, 59

P

ptext 91, 93

R

rt 47– 48, 54, 57, 62, 105

S

SRFACE 75
srfautoscal 75
srfhskirt 76

srfiskirt 76
srflabel 76
srfnpx 76
srfnpy 76
srftitle 76
srftopln 76
srfxhi 75
srfxlo 75
srfxtle 76
srfyhi 75
srfylo 75
srfytle 76
srfzhi 75
srfzlo 75
srfztle 76
stdplot 25, 91

T

textline 91
titleb 87, 102
titlel 87, 102
titler 87, 102
titlet 87, 102

U

ut 55, 62, 105

V

verbose 99
vt 55, 62, 105
vvgetc 65, 106
vvgeti 65, 106
vvgetr 65, 106
vvsetc 65, 106
vvseti 65, 106
vvsetr 65, 106

X

xmax 79–80
xmaxu 80
xmin 79–80
xminu 80

Y

ymax 79–80
ymaxu 80
ymin 79–80
yminu 80

Z

zt 47–48, 54, 57, 62, 105